APPLICAZIONI JAVA PARLANTI
SCOPRI TUTTE LE POTENZIALITÀ DELLA SINTESI VOCALE

VERSIONE PLUS ☑ RIVISTA+LIBRO+CD €9,90 — VERSIONE STANDARD ☐ RIVISTA+CD €6,90

ioPROGRAMMO

PER ESPERTI E PRINCIPIANTI

Poste Italiane • Spedizione in a.p. - 45% • art. 2 comma 20/b legge 662/96 - AUT. N. DCDC/033/01/CS/CAL Periodicità mensile • MARZO 2005 • ANNO IX, N.3 (89)

DENTRO BLUETOOTH
J2ME

IMPARA COME CREARE APPLICAZIONI PRONTE PER IL MONDO SENZA FILI E REALIZZA SUBITO UNA COMPLETA CHAT PER CELLULARI

- Scovare i dispositivi vicini abilitati
- Gestire la sicurezza e l'autenticazione
- Inviare un messaggio con foto

NOVITÀ
PROGRAMMARE IN PYTHON
Creare interfacce grafiche con questo nuovo rivoluzionario linguaggio di scripting

IOPROGRAMMO WEB
POSTA ELETTRONICA
Come inviare e ricevere email con C#

E-BAY
Crea un'asta sul tuo sito web sfruttando le API di E-Bay

VISUAL BASIC
INTERAGIRE CON XML
Un'applicazione per gestire calendari ed eventi

C++
VIDEOGAMES
Scontro fra mostri, ecco come gestire le collisioni

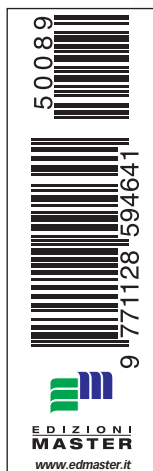
TECNICA
PROGRAMMA IL TUO GPS!
Gestire l'interfaccia seriale e la visualizzazione della posizione su una mappa

JAVA E MANTARAY
Un'applicazione per il controllo delle presenze con i Badge

PHP E JAVA
Phing e Hibernate: due metodi per gestire la persistenza dei dati. Come passare dal linguaggio SQL agli Oggetti!

.NET
SFRUTTARE AL MASSIMO LA CACHE
Come realizzare applicazioni velocissime utilizzando tutta la potenza di .NET

ADO.NET: TUTTE LE SOLUZIONI
Le risposte risolutive ai dubbi di programmazione sull'uso dei database

I CORSI PER IMPARARE
- VISUAL BASIC.NET 2003
- MACROMEDIA FLASH
- JAVA

DENTRO IL SISTEMA OPERATIVO: COME SFRUTTARE AL MEGLIO LA CPU

ioProgrammo Anno IX - N°3 (89)
Dentro Bluetooth • Persistenza dei dati con PHP e Java • Gestire la cache con .NET • ADO.NET: La guida definitiva
EDIZIONI MASTER

Anno IX - N.ro 3 (89) - Marzo 2005 - Periodicità: Mensile
Reg. Trib. di CS al n.ro 593 del 11 Febbraio 1997
Cod. ISSN 1128-594X
E-mail: ioprogrammo@edmaster.it
http://www.edmaster.it/ioprogrammo
http://www.ioprogrammo.it

Direttore Editoriale: Massimo Sesti
Direttore Responsabile: Massimo Sesti
Responsabile Editoriale: Gianmarco Bruni
MarCom Director: Luca Perfetti
Marketing Manager: Antonio Meduri
Editor: Gianfranco Forlino
Coordinamento redazionale: Raffaele del Monaco
Redazione: Fabio Farnesi
Collaboratori: M. Autiero, M. Bigatti, D. Bochicchio, L. Buono, F. Grimaldi, F. C. Ferracchiati, M. Locuratolo, A. Marroccelli, S. Meschini, G. Naccarato, G. Natili, F. Paparoni, P. Perrotta, L. Spuntoni, I. Venuti, F. Vaccaro.
Segreteria di Redazione Veronica Longo

Realizzazione grafica: Cromatika S.r.l.
Responsabile grafico: Paolo Cristiano
Coordinamento tecnico: Giancarlo Sicilia
Illustrazioni: M. Veltri
Impaginazione elettronica: Aurelio Monaco

“Rispettare l’uomo e l’ambiente in cui esso vive e lavora è una parte di tutto ciò che facciamo e di ogni decisione che prendiamo per assicurare che le nostre operazioni siano basate sul continuo miglioramento delle performance ambientali e sulla prevenzione dell’inquinamento”

Certificato UNI EN ISO 14001
N. 9191 CRMT

Realizzazione Multimediale: SET S.r.l.
Coordinamento Tecnico: Piero Mannelli
Realizzazione CD-Rom: Paolo Iacona

Pubblicità: Master Advertising s.r.l.
Via Ariberto, 24 - 20123 Milano
Tel. 02 831212 - Fax 02 83121207
e-mail advertising@edmaster.it
Sales Director: Max Scortegagna
Segreteria Ufficio Vendite: Daisy Zonato

Editore: Edizioni Master S.p.a.
Sede di Milano: Via Ariberto, 24 - 20123 Milano
Tel. 02 831212 - Fax 02 83121206
Sede di Rende: C.da Lecco, zona industriale - 87036 Rende (CS)
Presidente Amministratore Delegato: Massimo Sesti

ABBONAMENTO E ARRETRATI
ITALIA: Abbonamento Annuale: ioProgrammo Basic (11 numeri) €52,90 sconto 30% sul prezzo di copertina di €75,90 ioProgrammo con Libro (11 numeri + libro) €76,00 sconto 30% sul prezzo di copertina di €108,90
Offerte valide fino al 31/03/05
Costo arretrati (a copia): il doppio del prezzo di copertina + € 5.32 spese (spedizione con corriere). Prima di inviare i pagamenti, verificare la disponibilità delle copie arretrate allo 02 831212.
La richiesta contenente i Vs. dati anagrafici e il nome della rivista, dovrà essere inviata via fax allo 02 83121206, oppure via posta a EDIZIONI MASTER via Ariberto, 24 - 20123 Milano, dopo avere effettuato il pagamento, secondo le modalità di seguito elencate:

- cc/p n.16821878 o vaglia postale (inviando copia della ricevuta del versamento insieme alla richiesta);
- assegno bancario non trasferibile (da inviarsi in busta chiusa insieme alla richiesta);
- carta di credito, circuito VISA, CARTASI’, MASTERCARD/EUROCARD, (inviando la Vs. autorizzazione, il numero della carta, la data di scadenza e la Vs. sottoscrizione insieme alla richiesta).
- bonifico bancario intestato a Edizioni Master S.p.A. c/o Banca Credem S.p.a. c/c 01 000 000 5000 ABI 03032 CAB 80880 CIN Q (inviando copia della distinta insieme alla richiesta).

SI PREGA DI UTILIZZARE IL MODULO RICHIESTA ABBONAMENTO POSTO NELLE PAGINE INTERNE DELLA RIVISTA. L’abbonamento verrà attivato sul primo numero utile, successivo alla data della richiesta.
Sostituzioni: Inviare il CD-Rom difettoso in busta chiusa a:
Edizioni Master Servizio Clienti - Via Ariberto, 24 - 20123 Milano

Assistenza tecnica: ioprogrammo@edmaster.it

Servizio Abbonati:
tel.02 831212
e-mail: servizioabbonati@edmaster.it

Stampa: Rotoeffe Via Variante di Cancelliera, 2/6 - Ariccia (Roma)
Stampa CD-Rom: Neotek S.r.l.- C.da Imperatore - zona ASI Bisignano (CS)
Distributore esclusivo per l’Italia: Parrini & C S.p.A.
Via Vitorchiano, 81 - Roma

Finito di stampare nel mese di Febbraio 2005

Edizioni Master edita: Idea Web, Go!OnLine Internet Magazine, Win Magazine, Quale Computer, DVD Magazine, Office Magazine, La mia Barca, ioProgrammo, Linux Magazine, Softline Software World, HC Guida all’Home Cinema, MPC, Discovery DVD, Computer Games Gold, inDVD, I Fantastici CD-Rom, PC VideoGuide, I Corsi di Win Magazine, I Filmissimi in DVD, La mia videoteca, TV e Satellite, Win Extra, Home entertainment, Digital Japan, Digital Music, Horror Mania, Computer Bild, ioProgrammo Extra, Le Collection.

Non solo moda

In questo numero di ioProgrammo parliamo di Python. Alcuni di voi non avranno neanche mai sentito parlare di questo linguaggio, oppure lo avvertono come un eco lontano. In realtà Python non è ne una moda passeggera ne un argomento di poco conto. La stragrande maggioranze delle distribuzioni Linux lo utilizza per la progettazione degli script di mantenimento del sistema operativo, moltissimi tool professionali lo utilizzano allo stesso modo. Alcuni software specifici per la gestione di sistemi di Hosting ne fanno la base le loro sviluppo, ed è noto che l’hosting rappresenta un settore in enorme espansione e molto critico per quanto riguarda la sicurezza e la stabilità. Così abbiamo deciso di parlare di Python, perché riteniamo che la stragrande maggioranza degli utenti Windows non ne conoscano le potenzialità. Riteniamo infatti che il nostro compito sia non solo informare ma anche mostrare ai programmatori quali vie sono percorribili per produrre software di qualità e quali sono i settori che maggiormente in questo momento necessitano di personale competente. Python è uno di questi, persino in Google lo utilizzano molto frequentemente come base delle proprie applicazioni. Perciò vi invitiamo a leggere con attenzione il primo articolo con cui apriamo le porte a questo linguaggio, ad inviarci le vostre impressioni ad *ioprogrammo@edmaster.it* a partecipare al forum su *http://forum.ioprogrammo.net/board.php?boardid=37.*
Nonostante l’attenzione a Python, questo mese, come ogni mese, abbiamo tirato fuori dal cilindro alcune chicche davvero interessanti, si va da Bluetooth alla sintesi vocale passando per la persistenza dei dati con Java e PHP. Ce n’è per tutti i gusti, avrete sicuramente di che leggere e se progettate qualche applicazione interessante usando una delle tecniche illustrate non dimenticate di inviarla alla redazione, corredate di un mini articolo di spiegazione, le migliori potrebbero essere pubblicate nel CDROM allegato alla rivista.

Fabio Farnesi

All’inizio di ogni articolo, troverete un simbolo che indicherà la presenza di codice e/o software allegato, che saranno presenti sia sul CD (nella posizione di sempre *\soft\codice* e *\soft\tools*) sia sul Web, all’indirizzo http://cdrom.ioprogrammo.it.
Per scaricare software e codice da Internet, ogni mese indicheremo una password differente. Per il numero che avete fra le mani la combinazione è:

Username: **usr829** *Password:* **eight**

Dentro Bluetooth

J2ME

Crea un’applicazione per il toothing, il nuovo modo di fare conoscenze che sta dilagando in America e Inghilterra

- **Scovare i cellulari vicini abilitati**
- **Gestire la sicurezza e l’autenticazione**
- **Inviare un messaggio con foto**

pag. 16

APPLICAZIONI JAVA PARLANTI

http://forum.ioprogrammo.it

QUALCHE CONSIGLIO UTILE
I nostri articoli si sforzano di essere comprensibili a tutti coloro che ci seguono. Nel caso in cui abbiate difficoltà nel comprendere esattamente il senso di una spiegazione tecnica, è utile aprire il codice allegato all'articolo e seguire passo passo quanto viene spiegato tenendo d'occhio l'intero progetto. Spesso per questioni di spazio non possiamo inserire il codice nella sua interezza nel corpo dell'articolo. Ci limitiamo a inserire le parti necessarie alla stretta comprensione della tecnica.

RUBRICHE

ioProgrammo cerca articolisti freelance competenti nei seguenti argomenti:

Javascript, Python, Perl, ASP.NET, PHP, Flash, Security

Inviare curriculum dettagliato a ioProgrammo@edmaster.it

I contenuti del CD-Rom

AwStats 6.2
Le statistiche del tuo sito a portata di click
Directory: /awstats/awstats-62.exe

Apache 1.3.33/2.0.52
Uno dei server Web più usati al mondo
Directory: /Apache/

PHP 4.3.10/5.0.3
Il linguaggio di scripting necessario-per programmare il web
Directory /PHP

Python 2.3.4
Un linguaggio orientato agli oggetti con tanto di supporto a classi ed ereditarietà
Directory: /Python

Hibernate 2.1.7
Il framework per gestire la persistenza dei dati
Directory /Hibernate

MySQL 4.1.7
Il server di database OpenSource più diffuso al mondo
Directory /Mysql

PHPMyAdmin 2.6.0
Molto probabilmente il Frontend più usato al mondo per MySQL
Directory /PHPMyAdmin

Xmail 1.20
Il mailserver leggero è affidabile
Directory: /xmail

Wamp 5_1.4.3
Prova PHP+MySQL+Apache senza sforzo
Directory: /Wamp

Tortoise CVS
Per mantenere sotto controllo le variazioni apportate al codice
Directory: /TortoiseCVS

Sqlite 2.8.15
Il nuovo database Bundled con PHP5
Directory: /Sqlite

SharpDevelop 1.0.3.1761
L'alternativa a Visual Studio a basso costo
Directory: /Sharpdevelop

Mantaray 1.4.1
Java Messaging Service & PHP
Directory: /mantaray

Dev C++ 5 Beta 9
Un editor C++ a basso costo
Directory: /Devc++

JDK 1.5.0
Lo strumento indispensabile per sviluppare in Java
Directory: /jdk1.5.0

RIVISTA + CD-ROM in edicola

Prodotti del mese

JDK 1.5.0

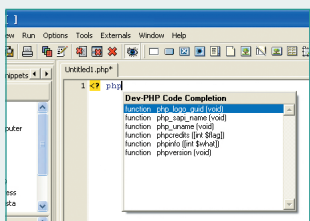
indispensabile per sviluppare in Java

Se siete sviluppatori Java o avete intenzione di imparare a programmare in Java avete sicuramente bisogno del JDK. In questo numero vi presentiamo la versione 1.5.0, rilasciata già da qualche mese e di cui ioProgrammo si sta occupando ampiamente in ogni numero.
La versione che vi proponiamo è particolarmente interessante, perché oltre al consueto JDK contiene in bundled una versione di NetBeans, ovvero un ambiente di programmazione appositamente studiato per applicazioni Java.
NetBeans è piuttosto potente anche se richiede un sistema con risorse adeguate.

Directory: /jdk1.5.0 [pag.116]

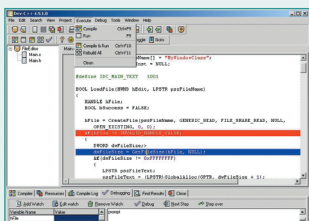
Dev-PHP 2.0.9

Ottimo editor PHP OpenSource

Se state iniziando a sviluppare in PHP avrete bisogno di un editor. Scrivere codice con il notepad può essere un esercizio divertente, ma quando iniziate a scrivere script leggermente più complessi si impone la scelta di passare a un editor più completo. DEV-PHP non solo è completo ma anche molto potente. Dotato di code completion, sintax highlighting, funzionalità di ricerca avanzate ed una serie di tool piuttosto interessanti rappresenta una grande scelta per programmare in PHP. Inoltre è un editor straordinariamente leggero, oltre che gratuito ed OpenSource.
Da non perdere!

Directory: /DevPHP [pag.116]

Dev C++ 5 Beta 9

Un editor C++ a basso costo

Dev C++ è un editor distribuito su licenza GPL, come tale non ha costi relativi al diritto d'autore. Come tutto o quasi tutto il software GPL la sua economicità non è affatto sinonimo di scarsa qualità. Al contrario Dev C++ è uno degli editor più amati ed utilizzati da chi sviluppa in C++. Le caratteristiche sono notevoli. Si va dal Debugger integrato, al project Manager, al Class Browser, al Code Completion.
L'insieme di queste caratteristiche, oltre una leggerezza innata dell'ambiente lo rende particolarmente comodo da utilizzare per sviluppare progetti C++ anche di grandi dimensioni.

Directory: /Devc++ [pag.115]

SharpDevelop 1.0.3

L'alternativa a Visual Studio

Microsoft senza dubbio ha modificato il suo modo di intendere lo sviluppo delle applicazioni con l'introduzione della piattaforma .NET.
Tuttavia lo sviluppo in tecnologia .NET non è una prerogativa dei soli ambienti targati Microsoft.
Esistono delle pur valide alternative a Visual Studio se si vuole comunque sviluppare in C# ad esempio oppure in VB.NET, SharpDevelop è una di queste. Visual Studio rimane sempre un ambiente estremamente completo colmo di caratteristiche che lo rendono unico, tuttavia SharpDevelop rappresenta un'ottima alternativa, economica, potente, affidabile.

Directory: /Sharpdevelop [pag.115

I contenuti del libro

RIVISTA + LIBRO + CD-ROM in edicola

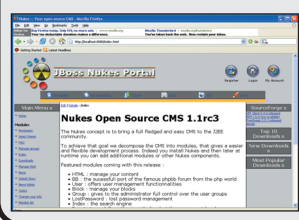
IMPARARE PHP

Un handbook di 150 pagine, rapido ed essenziale, privo di fronzoli ma incredibilmente stracolmo di esempi, è questo il nuovo libro di Fabio Farnesi che trovate allegato alla versione Plus di ioProgrammo. Imparare PHP è diviso in tre parti logiche, indipendenti e tali che chi non ha nessuna competenza di programmazione come chi invece è già un esperto può trovarne enormi vantaggi. Per un pubblico che si accinge ad iniziare la propria esperienza di programmazione in ambiente Web utilizzando PHP, il libro parte dalle basi, esaurendo l'argomento installazione, variabili, logica e strutture fondamentali per poi affondare nella programmazione ad oggetti. La seconda parte dell'handbook è dedicata alle estensioni di PHP ed in particolare a Pear e alla sua classe DB. Vengono introdotti i principi di interfacciamento con i database e soprattutto l'uso avanzato di Pear. La terza e ultima parte logica affonda il colpo esaurendo l'aspetto layout e in particolare mostra come separare la logica di programmazione da quella di progettazione dell'aspetto grafico di una web application utilizzando i template. Il motore scelto per questo tipo di tecnica è Smarty che sta rapidamente diventando un riferimento.

Prova subito

JBOSS NUKE PORTAL

Crea rapidamente il tuo Blog con JBoss e Java. Ed in più NetBeans l'editor che rende rapido lo

File su CD: /JBossnuke

Librerie e strumenti

ASP, C#, Java, Perl, VB.NET, Python, PHP, C++, XML

pag. 114

JBOSS Nuke Portal
Come PHP-Nuke ma in Java
Directory: /jbossnuke

JBOSS 1.4
L'application Server per applicazioni J2EE solide
Directory: /JBoss

Dev-PHP 2.0.9
Ottimo editor PHP OpenSource
Directory: /DevPHP

Hibernate 2.1.7
Il tool per la persistenza dei dati essenziale per programmare in JAVA.
Directory /Hibernate

Irrlicht 0.7
Sviluppare Giochi 3D potenti in modo semplice
Directory: /irrlicht

Jedit 4.2
Un editor per Java molto leggero e facile da usare
Directory: /jedit

Mod Asp Dot Net
Per programmare in .NET utilizzando APACHE
Directory: /mod_aspdotnet

Case Studio 2 2.18
Progettare diagrammi Entità-Relazione in modo semplice ed efficace
casestudio.zip

Tomcat 5.5.4
Il servlet container indispensabile per sviluppare il Web con Java e JSP
Directory: /tomcat

Repro 2.0
Una moviola per fermare il bug, facilita il test delle applicazioni
Repro-20.exe

MSDE Manager 3.09
Per gestire database MSDE attraverso una intuitiva interfaccia grafica
msde.exe

RC Localize 2.6
Ricerca le tue applicazioni all'interno dell'HD
RCLocalize.exe

EventStudio 2.5
Esplora i più complessi scenari con un potente tool CASE
eventstudio.zip

Install Creator 2.0 build 22
Per costruire installazioni facilmente
icinst.exe

ConversionTool 4.0 build 6
Importare database Access in SQL Server
DBX Conv.zip

Java Class Disassembler 4.0
Disassemblare file Java .class
jcd.zip

Census SB 6.0
Traccia i bug e i difetti dei tuoi progetti via Web
censussb.exe

C-Sharpener for VB 1.3
Basta un clic per tradurre da VB.NET a C#
c-sharpenerforvb.zip

EditLive for Windows 3.5
Integra l'authoring Web nei tuoi siti
editlivesdk.exe

Java Launcher 2.2
Lanciare applicazioni Java con un doppio clic
javalauncher.zip

Easy Java 1.2
Fai pratica con Java con un editor amichevole
^ezjava_setup.exe

UltraEdit-32 10.20d
HTML, testo, esadecimale: tutto in un editor
uedit32.zip

JDebugTool 3.8
Un debugger per Java full optional!
jdebugtool_sdk14.jar

Inno Setup 5.0.6
Installazioni professionali a costo zero
isetup-5.0.6.exe

JetBrains ReSharper 1.0.5
Un assistente per C# in Visual Studio
JetBrains.exe

Jurtle 1.7
Un ambiente di sviluppo integrato didattico per java
Jurtle_1.7.exe

GOOGLE ALL'ATTACCO DELL'HOSTING

A quanto pare Google sarebbe diventato un registar accreditato di ICANN e questo a meno di straordinari colpi di genio potrebbe voler dire semplicemente che Google intende proporre una sua offerta per la registrazione dei domini. Il mercato della registrazione dei domini è considerato strategico per attrarre nuovi utenti verso l'acquisto di servizi di Hosting e compagnie quali EV1Servers o GoDaddy ne hanno fatto un prezioso strumento di business. È anche vero che Google stima in $7.70 il costo per click di una campagnia pubblicitaria efficace mentre i domini vengono poi venduti ad una media di $6.50 per cui appare chiaro che la regsistrazione del dominio è un modo per attrarre i clienti verso servizi più costosi. Per quello che trapela al momento Google sarebbe in grado di registrare domini *.com, .net, .org, .biz.*, info, .name and .pro., non sappiamo davero cosa aspettarci dai vulcanici creativi di Google, ma sicuramente se questi rumors venissero confermati, la concorrenza di questo terribile competitors sarebbe da considerarsi decisamente pericolosa per le molte aziende che cercano di conquistare un posto al sole nel duro mercato dell'Hosting.

RILASCIATO VISUAL DATAFLEX 10.1

Visual DataFlex è il tool nato per sviluppare in modo rapido e visuale applicazioni che abbiano una forte integrazione con basi di dati. In realtà è garantito l'accesso a qualunque tipo di fonte rendendo di fatto l'indipendeza da un formato specifico. La nuova versione si presenta rivista e migliorata, dotata di una serie di wizard aggiuntivi che la rendono particolarmente semplice da usare e tuttavia ne lasciano inalterata la potenza. Utili anche le funzioni di "Desktop restore" che riportano il progetto alle condizioni in cui era stato lasciato al momento dell'ultima chiusura di DataFlex, e quelle relative alla creazione di report in formato PDF. A queste si sono aggiunte una serie di modifiche "minori" che rendono estremamente godibile l'uso del prodotto.

News

UN RFID NEGLI EURO?

Si fanno sempre più insistenti le voci che vogliono la banca centrale europea impegnata a cercare un fornitore di RFID per iniziare a stampare banconote che incorporino chip RFID. Il numero di chip necessario sarebbe così esorbitante da richiedere un pool di industrie, non esistendo nessuna singola azienda capace produrre volumi sufficienti. Se la notizia fosse confermata, comporterebbe una rivoluzione epocale: sarebbe estremamente più semplice tracciare la circolazione delle banconote e, anche la falsificazione riceverebbe un durissimo colpo.

Purtroppo nessun progresso può essere ritenuto positivo in assoluto. Anche in questo caso esistono, e sono pesantissimi, i lati negativi: gli acquisti in contanti sono l'ultimo baluardo per la difesa della privacy dei consumatori. Con la possibilità di tracciare ogni singola banconota, verrebbe meno anche quest'ultima difesa.

JAVA ALLA CONQUISTA DI IRC

È noto che la maggior parte dei canali IRC relativi ai vari circuiti internazionali di Chat siano regolati da Bot che altro non sono che un software altamente specializzato alla gestione dei canali. Nella stragrande maggioranza dei casi questi Bot sono basati su Eggdrop. Un software facilmente reperibile in rete all'indirizzo *http://www.geteggdrop.com/*, scritto in C++ estensibile nella maggior parte dei casi utilizzando il linguaggio TCL. Nel corso degli anni sono stati prodotti diversi altri tool pronti a dare l'attacco al vecchio ma sempre efficientissimo Eggdrop. Questa volta è il turno di un Bot scritto utilizzando completamente Java, il che lo rende particolarmente interessante. Drone, questo il nome del progetto, è reperibile all'indirizzo *http://drone.codehaus.org/*. Si tratta di tool completamente GPL e basato sul FrameWork RIFE *https://rife.dev.java.net/* del quale sentiremo ancora parlare in relazione allo sviluppo rapido di applicazioni per il Web.
Sembrerebbe che Drone sia in realtà un progetto sufficientemente consolidato, si legge infatti nelle varie dichiarazioni che per lunghi anni ha servito molti canali sulla rete freenode dove risiedono i canali irc della maggior parte dei progetti OpenSource oggi disponibili.

AL VIA LE E-MAIL CERTIFICATE

Il Consiglio dei Ministri ha approvato il decreto che sancisce l'assoluta parità fra e-mail certificate e posta ordinaria inviata con ricevuta di ritorno. L'Italia si colloca così fra i primissimi paesi al mondo ad adottare la mail come mezzo legale.
È bene chiarire che non tutte le mail avranno valore legale ma solo quelle dotate di un apposito certificato rilasciato da enti autorizzati dal ministero.
Grazie al decreto risultano regolamentate le due fasi fondamentali dello scambio di mail: il momento dell'invio e quello della ricezione.
In pratica, all'atto dell'invio, il mittente riceve una ricevuta che costituisce una prova legale dell'invio.
Parimenti, l'avvenuta consegna viene comunicata al mittente attraverso un messaggio anch'esso valido ai fini legali.
Il costo di questo servizio è prevedibile che sia di gran lunga inferiore a quello tradizionale e questo risparmio si andrà a sommare al risparmio ottenuto grazie all'assenza della carta e di tutta la relativa gestione.

VENTI ANNI DI NEWS INDICIZZATI DA GOOGLE

Google ha appena terminato l'indicizzazione di circa 800.000.000 messaggi comparsi in Usenet approssimativamente negli ultimi venti anni. Una cronistoria degli eventi che in un modo o nell'altro hanno segnato cambiamenti epocali annunciati nel corso del tempo in usenet è stata pubblicata a questo indirizzo *http://www.google.com/googlegroups/archive_announce_20.html*. Vi compare il primo messaggio con cui Torvalds annuncia la nascita di Linux, oppure la prima volta che il marchio Microsoft ha fatto il suo ingresso nel mondo dell'informatica, molto prima che Windows e persino Dos fossero concepiti.
Ma vi compaiono anche messaggi non squisitamente riguardanti temi informatici. Era il 1982 quando per la prima volta si è parlato di cellulari, nel 1983 la nota cantante "Madonna" ha fatto il suo ingresso nel mondo digitale, e nel 1985 si dibatteva su come affrontare lo spaventoso problema del Millennium Bug del 2000 che avrebbe dovuto segnare la morte dell'era informatica.
Il messaggio più bello è del sei agosto 1991 quando Tim Berners-Lee spiegava a grandi linee che stava nascendo un progetto World Wide Web indicandolo come un progetto per creare un semplice ma potente metodo per la creazione di un sistema informativo globale.

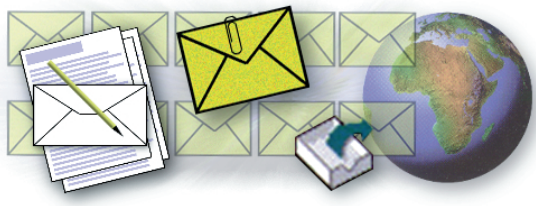
900.000 EURO DI DANNI RICHIESTI AD UN HACKER FRANCESE

Guillaume Tena meglio conosciuto con l'Aka Guillermito nel 2001 era un esperto di sicurezza. Nel corso delle sue ricerche individuò un certo numero di "vulnerabilità" in Viguard antivirus prodotto da Tegam e sulla base delle sue ricerche pubblicò vari exploit. La storia non fu di gradimento di Tegam che si lanciò in una battaglia legale. L'esito è stato sancito il 4 Gennaio. Quattro mesi di prigione sono stati decretati per Guilermito, inoltre dovrà pagare 6.000 euro di multa per violazione della legge sul copyright. Infine, quel che è peggio, è che è ancora in corsa una causa civile per danni nei riguardi di Tegam tale che Guillermito potrebbe dover sborsare la simpatica sommetta di euro 900.000 come risarcimento danni subiti da Tegam.
Ovviamente a questo primo grado di giudizio ne seguiranno altri, ed al momento il botta e risposta è elevato.

LIBRI SUPER ACCESSIBILI

L'iniezione di dollari ricevuta con la quotazione in borsa ha fatto di Google il più attivo laboratorio software che si sia mai visto. Ormai non si contano i servizi in fase sperimentale e alcuni di essi sono destinati a cambiare profondamente il rapporto degli utenti con Internet. La novità più chiacchierata (e temuta da qualcuno) è un nuovo browser, per costruire il quale pare siano stati già assunti due sviluppatori legati a Firefox. Ma non ci sono solo boatos, provate a collegarvi all'indirizzo video.google.com.
Vi aspetta la versione beta di un servizio che indicizza tutto quello che passa nelle TV americane. Ogni programma è registrato e viene archiviata la trascrizione di tutte le parole pronunciate durante la trasmissione. Le ricerche sono dunque possibili su tutti i dialoghi e su qualsiasi frase detta in TV. Provate a immaginare la quantità di contenuti che si riversa in questo modo in Internet! Ma non basta: Google ha lanciato un'altra iniziativa che avrà ricadute ancora più importanti. Il progetto Google Print ha lo scopo di digitalizzare il contenuto di milioni di libri, mettendone il contenuto a disposizione degli internauti. Ovviamente, fatti salvi i libri con diritti d'autore scaduti, sono da risolvere le notevole implicazioni legali di questa operazione che, una volta terminata, si potrà considerare come uno dei più grandi passi avanti nella diffusione della cultura. L'ultimissima novità vedrebbe Google impegnarsi sul fronte VoIP: è stato infatti pubblicato un annuncio in cui Google offriva lavoro a persone esperte nel campo delle "Dark fiber", ovvero i cavi a fibra ottica posati al tempo del boom della new economy e mai utilizzati.
Un operatore con le capacità finanziarie di Google potrebbe sfruttare questo sterminato groviglio di cavi inutilizzati per fornire servizi come quello di Skype, garantendo una qualità enormemente superiore. Insomma, i ragazzi di Google hanno soldi e idee: chi riuscirà a fermarli?

L'ANNO DEL PITONE

Vi sarete già accorti che anche noi di ioProgrammo stiamo cominciando a dare spazio a Python. In Italia questo linguaggio è ancora scarsamente diffuso, ma nel resto del mondo sta incontrando i favori di una nutritissima schiera di programmatori, tanto che tutte le statistiche lo danno come il linguaggio maggiormente in ascesa dell'ultimo anno. Questa volta ci occupiamo di Python in relazione alla più piccola applicazione P2P mai scritta. Appena 15 righe di codice Python compongono TynyP2P scritta da Edward Felten dell'università di Princetown in New Jersey nello sforzo di dimostrare che il P2P come strumento di condivisione decentralizzato di informazioni non può essere fermato.
Sempre in relazione all'ascesa di Python, fa bella mostra di se sul sito ufficiale www.python.org la notizia che riporta la disponibilità di "Python per i cellulari della serie Nokia 60". In pratica si può sviluppare in python anche per i cellulari!

Compilare PHP in ambiente Windows

Gentile redazione di ioProgrammo, anche grazie ai vostri articoli ho scoperto da poco il PHP e devo dire che lo trovo incredibilmente interessante. Mi sono divertito in ambiente Linux a ricompilare il codice adattandolo alle mie necessità, mi piacerebbe fare la stessa cosa in ambiente Windows con Visual Studio.NET ma ogni volta che tento una compilazione ricevo una marea di errori. Mi sapreste indicare una procedura corretta per la compilazione?

Cristian da Ascoli Piceno

Gentile Cristian
A prima vista una ricompilazione del codice di PHP in ambiente Windows sembrerebbe un'operazione superflua.
Di fatto tutte le DLL necessarie per estendere il linguaggio base con le funzioni aggiuntive sono contenute nella distribuzione binaria in formato *.zip*, ed attivabili semplicemente inserendo una riga di codice nel *php.ini*.
È anche vero che in molti casi la ricompilazione può risultare utilissima, ad esempio per aggiungere il supporto SSL a PHP oppure per inserire estensioni nuove programmate autonomamente. Abbiamo eseguito dei test in ambiente Windows XP utilizzando come compilatore Visual Studio.NET e anche noi abbiamo dovuto adottare qualche accorgimento per ottenere dei risultati soddisfacenti. In ogni caso la procedura da adottare è la seguente:

1) Scaricare i *Win32Build Tools* da *http://planetmirror.com/extra/win32build.zip*

2) Scaricare *DNSname Resolver* e da *http://planetmirror.com/com/bindlib_w32.zip*

3) Creare una directory *c:\work*

4) Scompattare *win32build* sotto *c:\work* di modo che sia contenuto nella cartella *c:\work\win32build*

5) Scompattare *BindLib* nella directory *c:\work* di modo che sia contenuto nella cartella *c:\work\bindlib_w32*

6) Scaricare l'ultima versione dei sorgenti di PHP5 ed estrarli nella directory *c:\work\php-5.0.x*

7) Creare una directory *c:\usr\local\lib* e copiare al suo interno il file *bison.simple* da *C:\work\win32build\bin*

8) A questo punto bisogna applicare una piccola patch al codice sorgente di PHP.
In particolare:

nel file *nsab_addr.c* contenuto in *C:\work\bindlib_w32* bisogna controllare che ci sia la seguente riga

```
#include "conf/portability.h"
```

e sostituire qualche riga più avanti

```
#if !defined(isxdigit) /* XXX - could be
                          a function */
```

con

```
#if !defined(isxdigit) && !defined(
                     _CTYPE_DEFINED)
    static int
    isxdigit(c)
register int c;
```

A questo punto siamo pronti per compilare

9) Avviare il prompt di Visual Studio dal menu *start* di Windows alla voce *"Visual Studio.NET 2003 prompt"*

10) Portarsi in *c:\work\php5.0.x* e lanciare

```
buildconf.bat
```

immediatamente dopo

```
cscript /nologo configure.js –help
```

vi comparirà la lista delle opzioni disponibili per la configurazione, ad esempio, le seguenti righe eliminano il supporto a *libxml* e a *iconv*

```
cscript /nologo configure.js
        --without-libxml --without-iconv
```

quando avrete deciso quali opzioni vi servono, digitare il comando e subito dopo digiare *n-make*.

Partirà la compilazione e, se tutto va a buon fine, troverete gli eseguibili nella directory *C:\work\php-5.0.3\Release_TS*

Accessibilità dei siti Web

Gentile redazione, sento spesso parlare di accessibilità dei siti web. Sono alle prime armi ma cerco di capirci qualcosa. Sto programmando il mio primo sito web e vorrei capire se ci sono dei parametri particolari da rispettare per realizzare qualcosa conforme alle norme.

Mi sapreste dire qualcosa in merito?

Claudio da Forlì

Gentile Claudio
la legge "Stanca" del 9 Gennaio 2004: *"Disposizioni per favorire l'accesso dei soggetti disabili agli strumenti informatici"*, regola la normativa relativa alla così detta "accessibilità dei siti web".
La legge trova fondamento nel riconoscimento e nella tutela da parte dello stato del diritto all'accesso ad ogni fonte informativa per qualunque persona. In questa ottica si pone una speciale attenzione a tutelare l'accesso all'informazione per le persone disabili.
La legge ha una particolare validità per i sistemi informativi, anche trasmessi per via telematica, della Pubblica Amministrazione.
In buona sostanza le Pubbliche Amministrazioni hanno, di fatto, l'obbligo di produrre siti web "accessibili" basati cioè sul regolamento attuativo della legge.
I privati hanno l'obbligo di produre siti accessibili se riconoscibili come aziende private concessionarie di servizi pubblici, aziende municipalizzate regionali, enti di assistenza e di riabilitazione pubblici, aziende di trasporto e di telecomunicazione a prevalente partecipazione di capitale pubblico, aziende appaltatrici di servizi informatici.
Nessun riferimento viene fatto ad altri soggetti all'interno della legge. Tuttavia è buona norma per ogni persona orientarsi verso la produzione di siti web accessibili.
Il regolamento attuativo della Legge, nella sua terza revisione, reca la data del 3 dicembre 2004.
Formalmente, i siti, per ottenere una specifica di conformità alle regole di accessibilità, devono soddisfare un criterio oggettivo basato su regole tecniche e un criterio soggettivo basato su un'analisi di qualità anche influenzato da considerazioni del tutto empiriche.
Per quanto riguarda i punti da rispettare per produrre siti "accessibili" (sono 22), ne citiamo solo qualcuno a scopo esemplificativo:

- È vietato l'uso dei Frame
- È necessario fornire una versione testuale per ogni oggetto non di testo presente nella pagina
- Garantire che i contenuti siano fruibili anche in assenza del colore che li contraddistingue
- Evitare oggetti o scritte lampeggianti la cui frequenza possa provocare disturbi da epilessia fotosensibile
- Usare grammatiche formali in particolare il riferimento è alla versione di Html 4.01 o Xhtml 1.0

Il testo integrale della legge 9 Gennaio 2004 è reperibile all'indirizzo *http://www.camera.it/parlam/leggi/04004l.htm*.
Il testo integrale del decreto attuativo è reperibile all'indirizzo *http://www.pubbliaccesso.it/biblioteca/documentazione/studio_lineeguida/index.htm*.

Apache e IIS sulla stessa porta

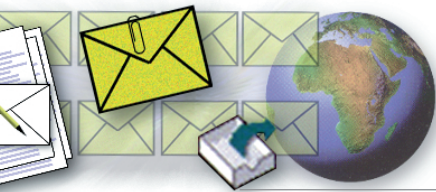
Desidero farvi i complimenti per le modifiche apportate agli ultimi numeri della rivista. Tutto risulta adesso molto più leggibile e facile da seguire anche se il livello degli articoli rimane alto. Tuttavia sono riuscito ad apprendere anche nozioni di argomenti di cui non ho una conoscenza strettissima grazie al nuovo modo di proporre gli articoli.
Veniamo al dunque.
Per quello che ne so dovrei poter fare girare sulla stessa porta più servizi se l'indirizzo IP è diverso.
Supponiamo che io abbia una macchina con due ip 192.168.0.2 e 192.168.0.3; in teoria dovrei potere fare girare due istanze di un mail server, ad esempio, sulle stesse porte ma bindandoli su due IP diversi. Effettivamente ho provato e funziona per tutto o quasi. Quando provo con Apache e IIS, non funziona mai, Apache si rifiuta di partire, non riesco a capire perché, c'è qualche intoppo in Windows che impedisce questa operazione, normalmente possibile in linea teorica?

Marcello da Palermo

Intanto siamo contenti che abbia apprezzato le modifiche che stiamo apportando alla rivista.
Vale per tutti la richiesta di inviarci feedback alla email *ioprogrammo@edmaster.it*, tutte le modifiche apportate sono frutto delle numerose segnalazioni che stiamo ricevendo.
È importante per noi fornire un servizio ai programmatori, tale da esservi utile nel vostro lavoro di ogni giorno.
Per quanto riguarda la sua domanda, effettivamente in linea teorica è possibile bindare IIS e Apache sulla stessa porta ma su IP differenti.
In realtà però IIS adotta il "Socket Pooling" tale che resta in ascolto su tutti gli IP anche quelli non bindati direttamente a un sito Web, per tale motivo Apache e IIS non possono girare sulla stessa porta enche se con IP differenti.
In IIS5 è ancora possibile disabilitare il Socket Pooling utilizzando l'utility Adsutil.vbs, in IIS6 questo invece non ha nessun effetto, di fatto questa funzionalità viene implementata a livello di Kernel in HTTP.sys e dovrebbe essere gestita tramite *Httpcfg.exe*.
La knolewdge base di Microsoft rispetto all'argomento è consultabile all'indirizzo *http://support.microsoft.com/kb/238131*.

Attenzione.
Disabilitare il Socket Pooling conduce a un decremento delle prestazioni.

PER CONTATTARCI

e-mail: ioprogrammo@edmaster.it
Posta: Edizioni Master,
Via Ariberto, 24 - 20123 Milano

Direttamente dai forum di ioProgrammo le discussioni più "Hot" del momento

Come implementare un calendario per un torneo

Quesito aperto
Discussione all'indirizzo
http://forum.ioprogrammo.net/thread.php?threadid=4047&boardid=13
di Ciufolo

Ciao a tutti, ho un problemino da affrontare. Vi spiego subito: devo costruirmi un piccolo sistema che riesca a creare un calendario di incontri. Sì... insomma come quelli che ci sono di calcio, basket... ecc. Dove bisognerà andare a recuperare le squadre all'interno di un DB. Però il problema è che, a parte l'utilizzo della proprietà Random, non ho molto chiare le idee su come poterlo realizzare, quindi sono qui per chiedere un consiglio a voi, se avete gi trattato cose di questo tipo, oppure se sapete indicarmi qualche buon link dove tratta di queste cose.
Grazie infinitamente in anticipo e un salutone.

Risponde Hyde
Fammi capire....
Tu hai le squadre memorizzate e devi generare il calendario (cioè gli incontri tra le squadre) in modo random o devi rispettare qualche criterio?
O devi fare qualcosa che non ho capito?!

Chiarisce Ciufolo:
Ciao Hyde, intanto grazie x avermi risposto più veloce della luce.
Riguardo al mio problema, hai capito benissimo, si i nomi delle squadre dovranno essere recuperate da DB dove vengono salvate al momento "dell'Iscrizione" (quando creo un nuovo giocatore), l'unico criterio che devo rispettare è che se *x* ed *y* si incontrano al 1° turno, ovviamente non si potranno incontrare anche nel 2°-3°-4°… turno fino alla fine di tutti gli accoppiamenti possibili.

Hanno risposto nell'ordine

Jachetto:
con una soluzione basata su XML in parte implemementata da Massimo Autiero e spiegata con un articolo in questo stesso numero di ioProgrammo

M.A.W 1968:
con una soluzione basata su calcolo combinatorio puro

VBLarge:
con un bell'esempio in Visual Basic

La discussione rimane ancora aperta. Chi troverà il metodo migliore?

Creare delle directory in Java

Di frank666
All'indirizzo
http://forum.ioprogrammo.net/thread.php?threadid=4398&boardid=18

Come si fa a creare una directory sul file system nel quale gira l'applicazione Java?
Grazie ciao

Risponde Krystal
Esistono due metodi della classe File:

- **mkdir** --> crea la dir rappresentata dall'istanza di File
- **mkdirs** --> crea la dir rappresentata dall'istanza di File e tutte quelle necessarie per realizzare il path
- --> entrambi i metodi restituiscono *true* se l'operazione è andata a buon fine. Per il secondo metodo considera però che potrebbe restituire false anche se è fallita solo la creazione di una delle directory del path specificato

Esempio

```
code:
boolean ok;
ok = new File("C:\pluto").mkdir();
ok &= new File("C:\paperino\paperina\
                              paperinick").mkdirs();
if(!ok)
...
```

la chiamata a *mkdirs* nell'esempio potrebbe creare la prima e la seconda directory *"paperino"* e *"paperina"* ma restituire *false* perché fallisce nella creazione di *"paperinik"*

Parserizzare un file di testo

Di d4n183 dal forum di ioProgrammo
http://forum.ioprogrammo.net/thread.php?threadid=4412&boardid=13

Ciao a tutti, è possibile fare in modo di leggere in un file di testo solo i caratteri compresi fra un particolare simbolo e trasformare esso in una stringa, esempio

ciao

posso fare in modo che venga preso solo ciao e ciò diventi una stringa?

Risponde Cteniza
Ti propongo qualcosa del genere:

```
Dim testo As String
testo = LeggiFile("c:\prova.txt")
Dim righe() As String
```

```
righe = Split(testo,vbCrLf)
Dim k As Long
Dim i As Integer
Dim c As String
c = "*"
Dim Campo As String
Dim Riga As String
For k = 0 To Ubound(righe)
Riga = Righe(k)
Do
i = InStr(1, Riga, c, vbTextCompare)
If i < 1 then Exit Do
Riga = Mid(Riga, i + 1)
i = Instr(1,Riga, c, vbTextCompare)
Campo = ""
if i > 1 then
Campo = Mid(Riga,1,i - 1)
If i < Len(Riga) then
Riga = Mid(Riga, i + 1)
else
Riga = ""
End If
else
Campo = Riga
Riga = ""
end if
Debug.Print Campo
Loop
Next
Function LeggiFile(nomeFile As String)
                                As String
Dim f As Integer
Dim s As String
f = FreeFile()
Open nomeFile For Binary Access Read As #f
s = Space$(LOF(f))
Get #f, , s
Close #f
LeggiFile = s
End Function
```

Rilasciato il .NET Compact Framework 1.0SP3

Di amdbook all'indirizzo http://forum.ioprogrammo.net/thread.php?threadid=4407&boardid=27

È uscito il SP3 per il .NET Compact Framework 1.0 al seguente link:

http://www.microsoft.com/downloads/details.aspx?displaylang=it&FamilyID=a5a02311-194b-4c00-b445-f92bec03032f
come da documentazione, difetti eliminati:

- Possibili perdite di memoria nelle transizioni da codice gestito a codice nativo sulle piattaforme ARM
- Eccezione *NullReferenceException* generata quando un metodo Web restituisce un array vuoto con l'attributo xsi:Nil
- Mancata modifica dell'intestazione ContentType delle richieste HTTP dopo aver modificato la proprietà *SoapClientMessage.ContentType*
- Possibile errore dello stack sulle piattaforme SH, MIPS e x86 quando vengono create ma mai utilizzate le variabili locali
- Eccezione *MissingMethodException* generata sulle piattaforme SH, MIPS e x86 quando un delegato multicase è richiamato dal gestore catch
- Troncamento a un solo byte degli argomenti della riga di comando contenenti caratteri a due byte
- Eccezione *ObjectDisposedException* generata quando viene annullata una richiesta Web asincrona prima della ricezione della risposta
- Blocco dell'applicazione in seguito al richiamo di un controllo eliminato
- Invio non corretto di array contenenti uno o più elementi al servizio Web
- Possibile blocco di un'applicazione al richiamo di un metodo Web con più attributi *XmlElementAttribute* in un solo argomento, membro o proprietà
- Possibile danneggiamento della memoria sui dispositivi in cui è abilitato il modello di protezione nativo e sia .NET CF V1 SP3 che una versione preliminare di .Net CF V2 installata.
- Possibile blocco critico (deadlock) durante l'esecuzione di vincoli critici di risorse
- Immanenza delle immagini della barra degli strumenti in Windows Mobile 2003 SE anche dopo la rimozione dal form
- Eccezione *ObjectDisposedException* non rilevabile generata quando il server chiude la connessione del socket
- Arresto anomalo dell'applicazione eliminato con l'impostazione delle proprietà Minimum e Maximum di un controllo *Progressbar*

Javascript e i cicli

Da s8i4 http://forum.ioprogrammo.net/thread.php?threadid=4571&boardid=5

Quando ho trovato un risultato prima che finisca il ciclo come faccio a uscirne fuori? Ho provato con while ma non riesco

Risponde berta_danilo
Devi usare l'istruzione break, come nell'esempio.

```
<script type="text/javascript">
var idx;
for(idx=1;idx<=30;idx++)
{
  document.write("Il numero " + idx +
                                   "<br>");
  if(idx == 20)
     break;
}
</script>
```

Spero di esserti stato utile.

Realizzare un programma per il Toothing dal nostro cellulare

Chat Bluetooth

Il Toothing, ovvero una moda dilagante in Inghilterra e America che consente di trovare nuovi amici chattando dal cellullare a costo zero. Creiamo un'applicazione in J2ME

Il Bluetooth è una tecnologia per la connessione wireless tra diversi dispositivi. Attualmente viene utilizzato per auricolari senza fili, per creare delle reti sul momento, per trasferimento di file e per molte altre cose. Esistono anche altre tecnologie per questo tipo di collegamento, come infrarossi e wireless 802.11b, ma il chip Bluetooth è quello che ultimamente viene inserito in quasi tutti i dispositivi mobili come cellulari e palmari e che quindi ha una maggiore diffusione. Questi dispositivi hanno quasi sempre a bordo un ambiente di esecuzione per applicazioni J2ME, quindi è interessante vedere come sia possibile pilotare con un programma Java una connessione di questo tipo.

FUNZIONAMENTO

Prima di tutto dobbiamo capire come funziona il collegamento tra due dispositivi Bluetooth. Ci sono diverse fasi attraverso le quali possiamo strutturare tutto il funzionamento. La prima fase è quella della ricerca. Il dispositivo Bluetooth che vuole iniziare una comunicazione deve effettuare una ricerca per trovare gli altri dispositivi raggiungibili. Una volta che sono stati trovati, il dispositivo decide con quale mettersi in comunicazione e quindi inizia la ricerca dei servizi disponibili su un determinato device.

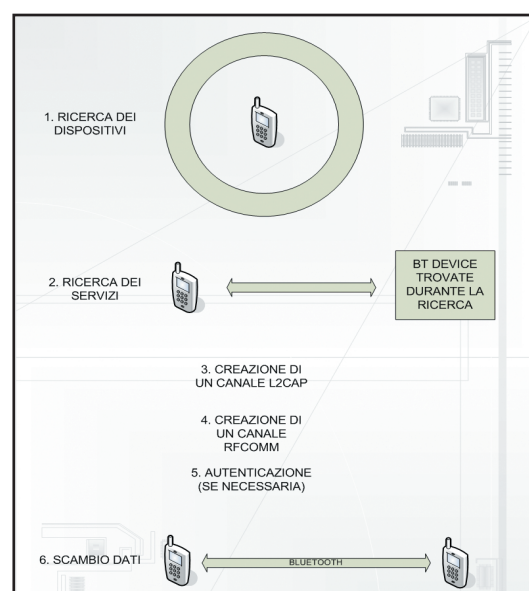
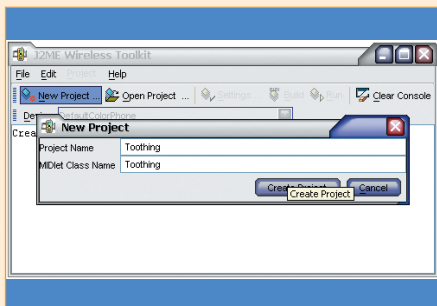
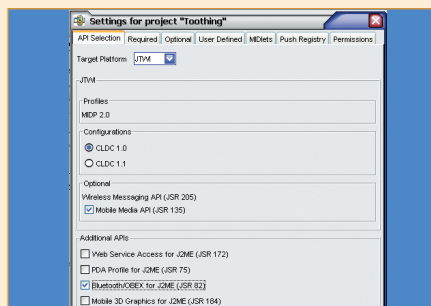
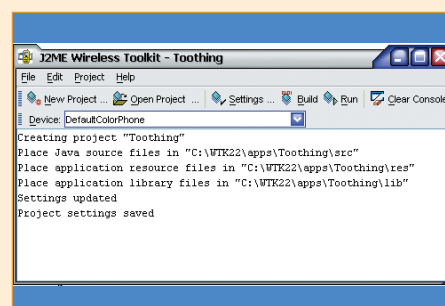



Fig. 1: *Le diverse fasi della comunicazione Bluetooth*

COME INIZIARE

CREA UN NUOVO PROGETTO

1 Prima di tutto bisogna avviare Ktoolbar, l'ambiente di sviluppo per J2ME fornito dalla SUN.
Cliccando sul bottone "New Project" creiamo un nuovo progetto, per il quale dobbiamo inserire il nome.

SETUP DEL PROGETTO

2 Procediamo quindi con il setup della nostra applicazione. Per compilare un programma per un determinato dispositivo dobbiamo sapere che configurazione J2ME ha e quindi settare i corrispondenti valori nel nostro ambiente.

CARTELLE DEL PROGETTO

3 Settati tutti i parametri, il Toolkit crea un cartella specifica. Nella sottocartella "*src*" inseriremo i sorgenti, in "*res*" immagini e file che richiameremo e in "*lib*" librerie aggiuntive. In "*bin*" troveremo i file compilati per il cellulare.

Una volta selezionato un servizio viene instaurata una connessione usando il protocollo *L2CAP (Logical Link Control and Application Protocol)*. Sopra a questo protocollo, che gestisce direttamente i dati per adattarli al formato di comunicazione per il Bluetooth, può essere utilizzato un protocollo simile al *RFCOMM (Radio Frequancy Communications Protocol)* che simula la comunicazione con una porta seriale. A questo punto è necessario autentificarsi con l'altro dispositivo tramite l'invio di un PIN che viene richiesto dal dispositivo che mette a disposizione il servizio. Se il PIN inviato è corretto inizia il vero e proprio scambio di dati tra i due dispositivi, che dipendono chiaramente da come sono strutturate le applicazioni che comunicano.

J2ME E JSR82

Per quanto riguarda la versione MicroEdition di Java sono stati sviluppati diversi package opzionali. Uno di questi è quello che riguarda la comunicazione Bluetooth. La JSR82 definisce le Java Bluetooth API, con due package che possono essere utilizzati nella realizzazione di applicazioni J2ME che dialogano con un dispositivo Bluetooth. Il primo package, *javax.bluetooth*, è la parte fondamentale di queste librerie il quale permette di realizzare la comunicazione standard che è stata prima descritta. Il secondo package, *javax.obex*, permette l'utilizzo del profilo *OBEX* (*Object Exchange*) in applicazione J2ME, ovvero poter serializzare e deserializzare dei veri e propri file attraverso la comunicazione Bluetooth. Grazie quindi a questo package aggiuntivo è possibile creare delle applicazioni che comunichino con la radio Bluetooth del nostro cellulare, avendo a disposizione un vero e proprio stack Bluetooth. Non essendo un profilo standard come MIDP, non troveremo le Bluetooth API su tutti i dispositivi J2ME (specialmente in cellulari datati). D'altronde rispetto a qualche mese fa sono disponibili molti cellulari che hanno un'implementazione della JSR82 e, l'interesse verso la programmazione di applicazioni che utilizzano questo tipo di tecnologia, è chiaramente cresciuto negli ultimi tempi.

TOOTHING: UN ESEMPIO DI APPLICAZIONE

Addentriamoci quindi nello sviluppo di una semplice applicazione, utilizzando le BT API. Il termine toothing definisce una vera e propria moda ultimamente in voga. È un modo alternativo per fare nuove amicizie e si basa appunto su dispositivi Bluetooth. Praticamente, questa tecnologia si usa lasciando attivo il Bluetooth sul cellulare, poi per ricercare nuove amicizie si ricercano altri dispositivi in zona e se si trovano viene inviato un semplice messaggio o una foto per iniziare una conversazione. Per incominciare a giocare con queste BT API possiamo pensare di fare un'applicazione per il Toothing dal cellulare. L'architettura che vogliamo realizzare è abbastanza semplice ed è composta da una sola applicazione, che può svolgere le funzioni di server o di client. Allora, dal punto di vista del server dobbiamo mettere a disposizione di eventuali client un servizio Bluetooth che riceverà le informazioni inviate dal client e le notificherà all'utente. In maniera simmetrica il client

La tecnologia Bluetooth opera sulla frequenza dei 2,4 Ghz e permette una comunicazione tra dispositivi intorno ai 1 Mbit/s. Esistono 3 diverse classi di funzionamento per quanto riguarda i chip Bluetooth, che si differenziano per la potenza in uscita del dispositivo e quindi il raggio d'azione. La classe 1 va dai 50 ai 120 metri, la classe 2 dai 10 ai 30 e la terza da 1 a 10

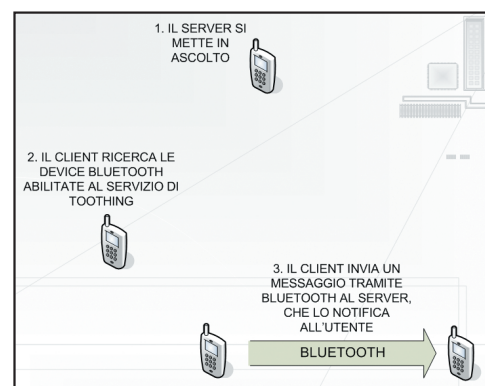
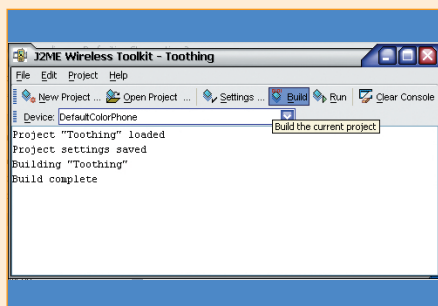
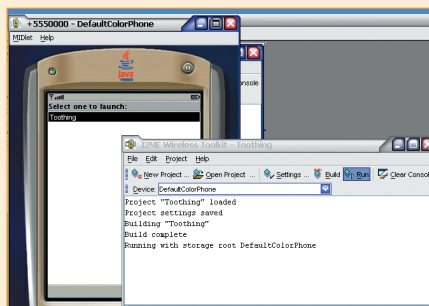
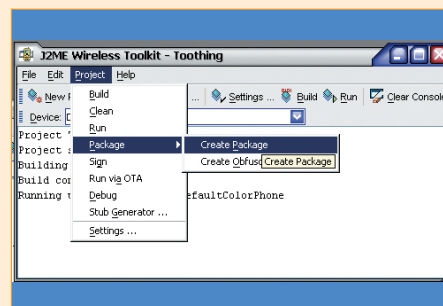



***Fig. 2:* Architettura dell'applicazione**

COMPILAZIONE

4 **Una volta editati i file della nostra MIDlet, dobbiamo inserirli nella cartella "src" e premere il pulsante "Build" di Ktoolbar. In questo modo il nostro progetto viene compilato e nel caso ci sia qualche problema la compilazione fallisce con un apposito messaggio d'errore.**

ESECUZIONE

5 **Completata la compilazione possiamo effettuare un test facendo partire l'emulatore, con il bottone "*Run*" che mostrerà un cellulare con dentro l'applicazione. Sarà necessario avviare due diversi emulatori con la stessa applicazione.**

CREARE IL PACKAGE

6 **Per trasferire l'applicazione su cellulare dobbiamo creare i file JAD e JAR, accedendo al menu *Project > Package > Create Package*, che crea i due file necessari nella sottocartella "*bin*". In seguito li invieremo via Bluetooth, IRDA o OTA al cellulare ed installeremo l'applicazione.**

dovrà ricercare le device Bluetooth, selezionare i servizi disponibili e inviare delle informazioni configurate in partenza. La parte più complicata sarà quindi dal punto di vista del client, visto che per quanto riguarda il server non dobbiamo far altro che pubblicare il nostro servizio e rimanere in attesa.

DEVICE DISCOVERY

Il nostro client dovrà prima di tutto ricercare delle device attive, nel raggio d'azione del Bluetooth. Per utilizzare le BT API dobbiamo implementare l'interfaccia *DiscoveryListener* nella classe che sarà la responsabile per le comunicazioni BT della nostra applicazione.

Method Summary	
void	**deviceDiscovered**(RemoteDevice btDevice, DeviceClass cod) Called when a device is found during an inquiry.
void	**inquiryCompleted**(int discType) Called when an inquiry is completed.
void	**servicesDiscovered**(int transID, ServiceRecord[] servRecord) Called when service(s) are found during a service search.
void	**serviceSearchCompleted**(int transID, int respCode) Called when a service search is completed or was terminated because of an error.

Fig. 3: L'interfaccia DiscoveryListener

Questa interfaccia richiede l'implementazione di 4 metodi:

1. **deviceDiscovered(RemoteDevice btDevice, DeviceClass cod):** Questo metodo viene richiamato quando durante una ricerca viene trovata una device.
2. **inquiryCompleted(int discType):** Quando la ricerca è terminata e il numero che viene passato come parametro descrive se la ricerca è stata completata o è terminata per altre cause.
3. **servicesDiscovered(int transID, ServiceRecord [] servRecord):** Vengono restituiti un array di servizi trovati durante la ricerca e un ID della ricerca che ha recuperato questi risultati.
4. **serviceSearchCompleted(int transID, int respCode):** Richiamato alla fine della ricerca di un servizio, con il relativo ID e il modo in cui è terminata.

Il nome Bluetooth deriva dal re vichingo Harald Bluetooth che unificò la Norvegia alla Danimarca sotto un unico regno. Viene ricordato appunto per questa impresa, per l'unificazione di due popoli totalmente diversi, e per questo motivo questa tecnologia wireless ha preso il suo nome.

Ora per inizializzare la parte del client che effettuerà la ricerca dobbiamo istanziare un *DiscoveryAgent*, ovvero la classe che effettuerà le ricerche e definire un numero massimo di ricerche di servizi

```
LocalDevice local = LocalDevice.getLocalDevice();
DiscoveryAgent agent = local.getDiscoveryAgent();
//Dopo aver ottenuto un DiscoveryAgent settiamo
//il numero massimo di ricerche da effettuare
//ricorrendo ad una proprietà della BT API
try {
  maxServiceSearches = Integer.parseInt(
  LocalDevice.getProperty("bluetooth.sd.trans.max"));}
catch( NumberFormatException e ) {
    System.out.println( "General Application Error" );
    System.out.println( "NumberFormatException: " +
                                    e.getMessage() );}
//Infine creiamo un array di transactionID e li
                                    inizializziamo a -1
transactionID = new int[maxServiceSearches];
for( int i=0; i<maxServiceSearches; i++ ) {
  transactionID[i] = -1; }
record = null;
deviceList = new Vector();
```

La ricerca delle device Bluetooth avviene in diverse modalità. Infatti, prima di iniziare una vera e propria ricerca attivando il BT possiamo cercare nel nostro dispositivo delle device remote che sono state precedentemente messe in cache o che sono già conosciute dal nostro sistema. Se non abbiamo queste device allora inizia la vera e propria ricerca di dispositivi, grazie alla quale ci viene restituito un *ServiceRecord*, descrizione di una caratteristica di un servizio BlueTooth.

```
RemoteDevice[] devList = agent.retrieveDevices(
                              DiscoveryAgent.CACHED );
if( devList != null ) {
  if( searchServices(devList) ) {
      return record; } }
devList = agent.retrieveDevices(
                           DiscoveryAgent.PREKNOWN );
if( devList != null ) {
  if( searchServices(devList) ) {
     return record; } }
try {
       agent.startInquiry(DiscoveryAgent.GIAC, this);
       synchronized( this ) {
          try {this.wait();}
          catch (Exception e) {}  }  }
    catch( BluetoothStateException e ) {
       System.out.println( "Impossibile ricercare device" ); }
    if( deviceList.size() > 0 ) {
      devList = new RemoteDevice[deviceList.size()];
      deviceList.copyInto( devList );
      if( searchServices(devList) ) {
   return record; } }
```

Il ServiceRecord che otteniamo contiene delle informazioni relative al servizio Bluetooth e in base ad

Attribute Name	Attribute ID	Attribute Value Type
ServiceRecordHandle	0x0000	32-bit unsigned integer
ServiceClassIDList	0x0001	DATSEQ of UUIDs
ServiceRecordState	0x0002	32-bit unsigned integer
ServiceID	0x0003	UUID
ProtocolDescriptorList	0x0004	DATSEQ of DATSEQ of UUID and optional parameters
BrowseGroupList	0x0005	DATSEQ of UUIDs
LanguageBasedAttributeIDList	0x0006	DATSEQ of DATSEQ triples
ServiceInfoRimeToLive	0x0007	32-bit unsigned integer
ServiceAvailability	0x0008	8-bit unsigned integer
BluetoothProfileDescriptorList	0x0009	DATSEQ of DATSEQ pairs
DocumentationURL	0x000A	URL
ClientExecutableURL	0x000B	URL
IconURL	0x000C	URL
VersionNumberList	0x0200	DATSEQ of 16-bit unsigned integers
ServiceDatabaseState	0x0201	32-bit unsigned integer

Tabella 1: ID standard per un servizio Bluetooth

esso possiamo avere informazioni come il nome, l'url della documentazione e tante altre che descrivono il servizio che abbiamo incontrato.

SERVICE DISCOVERY

Siamo ora arrivati al punto di dover cercare i servizi disponibili su ogni singola device. Prima di tutto dobbiamo definire degli *UUID (Universally Unique IDentifiers)* che andremo a cercare.

```
UUID[] searchList = new UUID[2];
searchList[0] = new UUID( 0x0100 );
searchList[1] = new UUID(
      "00112233445566778899AABBCCDDEEFF", false);
```

In questo modo abbiamo definito che siamo interessati ai servizi che hanno come UUID 0x0100 e 00112233445566778899AABBCCDDEEFF. Questo serve per poter scegliere direttamente i servizi che ci vogliamo trovare, infatti il lungo UUID, che evito di ripetere, definisce l'identificatore per il nostro servizio. Ora dobbiamo semplicemente chiedere al nostro DiscoveryAgent, in un ciclo for, di ricercare l'array di UUID e salvare l'ID di transazione in una tabella.

```
try
{
    System.out.println( "Ricerca dei servizi su " +
                        devList[i].getBluetoothAddress() );
    int trans = agent.searchServices( null,
                        searchList, devList[i], this );
    System.out.println( "ID transazione " + trans );
    addToTransactionTable( trans );
}
catch( BluetoothStateException e )
{
   System.out.println( "BluetoothStateException: " +
                        e.getMessage() );
}
```

Quando vengono trovati dei servizi viene richiamato il metodo *servicesDiscovered* dell'interfaccia *DiscoveryListener*. In questo metodo possiamo stampare a schermo i servizi trovati. Ora l'importante è sapere quale sia l'URL del servizio che ci interessa per poter comunicarci direttamente. Questa informazione la possiamo avere soltanto dal ServiceRecord che ci è stato restituito, attraverso il metodo getConnectionURL.

```
String conURL = echoService.getConnectionURL(
 ServiceRecord.NOAUTHENTICATE_NOENCRYPT, false );
```

A questo punto abbiamo, dal punto di vista del client, l'indirizzo con il quale possiamo comunicare.

BINDING DEL SERVIZIO

Come già detto, dal punto di vista del server il lavoro da fare è molto minore. Dobbiamo semplicemente settare il nostro device raggiungibile in Bluetooth dalle altre periferiche. Poi dobbiamo aprire un classico Connector su un indirizzo che è composto dall'UUID che abbiamo scelto per la nostra applicazione e aspettare una connessione

```
LocalDevice local = LocalDevice.getLocalDevice();
local.setDiscoverable( DiscoveryAgent.GIAC );
Server = (StreamConnectionNotifier)Connector.open(
           "btspp://localhost:00112233445566778899
                              AABBCCDDEEFF" );
conn = (StreamConnection)Server.acceptAndOpen();
BlueListener.Client = new ClientProcess(conn,bs,this);
BlueListener.Client.start();
```

Una volta arrivata una connessione dovremo avviare un thread separato che si occuperà di ricevere le informazioni inviate dal client.

IL PROTOCOLLO

Per trasferire le informazioni da client a server dobbiamo decidere un protocollo da utilizzare. Infatti le nostre informazioni arriveranno al server come un flusso di byte e quindi stabilendo un nostro protocollo possiamo semplicemente serializzare e deserializzare le informazioni. In questa applicazione d'esempio vengono inviate tre diverse informazioni, un'immagine, un nickname e un messaggio. Possiamo quindi strutturare il nostro protocollo mettendo come header iniziale 4 byte di informazione sull'intero pacchetto. Nei primi due byte inseriamo la lunghezza della nostra immagine, facendo prima il modulo e poi la divisione della sua lunghezza (altrimenti dovremmo mandare delle immagini piccole). Negli altri due byte metteremo la lunghezza del nick e del messaggio. In questo modo il server leggerà i primi 4 byte del flusso di dati e in base a questi continuerà a leggere fino a quando riempirà i buffer rispettivamente per l'immagine, il nick e il messaggio.

```
//Lato client
byte[] info=new byte[4];
info[0]=(byte)(foto.length%256);
info[1]=(byte)(foto.length/256);
info[2]=(byte)(mess.getBytes().length);
info[3]=(byte)(nick.getBytes().length);
op.write(info);
op.flush();
op.write(foto);
```

Il Bluetooth SIG (Special Interest Group) è un consorzio per lo sviluppo di applicazioni basate sulla tecnologia Bluetooth. I maggiori promotori di questo gruppo sono aziende come Agere, Ericsson, IBM, Intel, Microsoft, Motorola, Nokia, e Toshiba.
http://www.bluetooth.com/

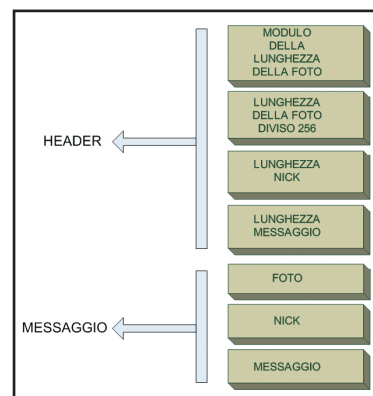



Fig. 4: Struttura del pacchetto utilizzato nell'applicazione

```
op.flush();
op.write(mess.getBytes());
op.flush();
op.write(nick.getBytes());
op.flush();
connection.close();
```

Ora il server riceve queste informazioni e crea dinamicamente array di byte per contenere le informazioni.

```
//Lato server
byte[] info=new byte[4];
is.read(info);
modulo=info[0]+256;
div=info[1];
grandezza=(div*256)+modulo;
byte c;
//Ricezione foto
for (int i=0;i<grandezza;i++) {
    c=(byte)is.read();
    immago[i]=c; }
grandezza=info[2];
mess=new byte[grandezza];
//Ricezione messaggio
for (int i=0;i<grandezza;i++) {
   c=(byte)is.read();
   mess[i]=c;
}
grandezza=info[3];
nick=new byte[grandezza];
//Ricezione nick
for (int i=0;i<grandezza;i++) {
    c=(byte)is.read();
    nick[i]=c; }
```

Per modificare questo protocollo, magari pensando ad altri tipi di informazione che vengono passati dovremmo solo rivedere l'header iniziale nel quale inseriremmo nuovi campi.

SUL WEB

Per scrivere le nostre applicazioni J2ME abbiamo bisogno prima di tutto di aver installato la versione SE di Java sul nostro computer, gratuitamente scaricabile dal sito della SUN.
http://java.sun.com/j2se/1.4.2/download.html

Poi dobbiamo avere un emulatore per testare le nostre applicazioni. Sempre sul sito della SUN troviamo il Wireless Toolkit per J2ME (http://java.sun.com/products/j2mewtoolkit/download-2_2.html)

In alternativa possiamo utilizzare l'emulatore della Nokia, scaricabile gratuitamente da www.forum.nokia.com.

INSTALLAZIONE DELL'APPLICAZIONE

Le applicazioni J2ME possono essere installate sui cellulari/palmari in diversi modi. Quando compiliamo la nostra MIDlet abbiamo a disposizione due diversi file, il file JAD e il JAR. Il file JAD *(Java Application Descriptor)* descrive la nostra applicazione attraverso certi attributi, mentre il JAR*(Java Archive)* è il vero e proprio programma. Possiamo utilizzare un deployment dell'applicazione OTA *(Over The Air)*, ovvero fare l'upload della nostra applicazione su un Web Server (che abbia configurato il content type relativo ai file JAD e JAR) e puntare l'indirizzo dal browser WAP del nostro cellulare. In alternativa possiamo passare le applicazioni tramite i cavi di connessione USB o seriali del dispositivo. Eventualmente è possibile inviare le applicazione anche tramite la porta infrarossi o una chiave Bluetooth. Di solito comunque viene fornito un CD con il software originale per il nostro cellulare, una sorta di pc suite che permette, oltre alla sincronizzazione di rubriche e messaggi, anche l'invio di MIDlet.

LA MIDLET

Nella nostra MIDlet dobbiamo poter scegliere la modalità da utilizzare, ovvero in ricezione o in invio. La modalità di ricerca avvierà la parte server e rimarrà in ascolto di messaggi. Una volta arrivato un messaggio lo deserializzarà come già illustrato e lo notificherà all'utente. Per quanto riguarda la modalità di invio dovremo permettere all'utente di inserire le informazioni riguardanti il nick e il messaggio da inviare.

```
principale= new Form("Invio Messaggio");
d = Display.getDisplay(this);
utente=new TextField("Utente:", "",10, TextField.ANY);
note=new TextField("Messaggio:", "",40, TextField.ANY);
invia=new Command("Invia", Command.OK, 1);
esci=new Command("Esci", Command.EXIT, 2);
principale.addCommand(invia);
principale.addCommand(esci);
principale.setCommandListener(this);
principale.append(utente);
principale.append(note);
d.setCurrent(principale);
```

Una volta che l'utente ha riempito i TextField, prendiamo una generica immagine che abbiamo inserito all'interno della nostra MIDlet e facciamo partire un thread che gestirà tutta la parte riguardante il Device Discovery, il Service Discovery e l'invio delle informazioni.

```
String nick=utente.getString();
String msg=note.getString();
BlueSender bs=new BlueSender(
                    this,msg,nick,openFile("uni.png"));
bs.start();
```

Dal punto di vista del server invece dobbiamo implementare un listener che notifica alla nostra interfaccia grafica l'arrivo di un nuovo messaggio. Una volta notificato dobbiamo rappresentare il messaggio arrivato sul display del cellulare.

Fig. 5: L'applicazione testata nel Wireless Toolkit

Fig. 6: L'applicazione testata nel Wireless Toolkit

```
public void leggimess() {
principale= new Form("Messaggio Arrivato");
  d = Display.getDisplay(this);
  continua=new Command("Continua", Command.OK, 1);
  esci=new Command("Esci", Command.EXIT, 2);
  principale.addCommand(continua);
  principale.addCommand(esci);
  principale.setCommandListener(this);
  String tempnick=listener.getMsgNick();
  String tempmess=listener.getMsgMess();
  Image tempimmago=listener.getMsgImage();
  StringItem tempitem=new StringItem("Messaggio
                    da "+tempnick,"Testo: "+tempmess);
  principale.append(tempitem);
  principale.append(tempimmago);
  nuova=false;
  d.setCurrent(principale); }
```

L'engine vero e proprio che ricerca le informazioni via Bluetooth e che le invia dovrà implementare, come già detto l'interfaccia DiscoveryListener definita nel package javax.bluetooth. Chiaramente i metodi che attivano la ricerca e l'invio dei messaggi è opportuno separarli dall'interfaccia grafica per evitare un blocco dell'applicazione, essendo le chiamate delle BT API bloccanti per l'applicazioni. Quindi ci sarà un thread che gestisce i messaggi in uscita (client) e uno per i messaggi in entrata (server).

TESTIAMO L'APPLICAZIONE

Per testare la nostra applicazione dobbiamo avere a disposizione il Wireless Toolkit della SUN, versione 2.2, grazie al quale possiamo implementare e testare le applicazioni che fanno uso delle BT API, oltre ad altri package aggiuntivi. L'emulatore fornito dalla SUN permette di testare le applicazioni Bluetooth simulando la connessione tra diversi dispositivi. Mentre di solito abbiamo bisogno di un solo emulatore per testare le applicazioni J2ME, in questo caso dobbiamo avviare due diversi emulatori per avviare su uno l'applicazione in modalità client, sull'altro in modalità server. Una volta aperto il Wireless Toolkit, apriamo il nostro progetto e selezioniamo per due volte Run, in questo modo vengono caricati due diversi emulatori con la stessa MIDlet. Nel primo emulatore selezioniamo attraverso il menu la ricerca. Ora questo emulatore espone un servizio Bluetooth e rimane in attesa di un client. Sull'altro emulatore facciamo partire la modalità invio e una volta inserito il nick e il messaggio da inviare selezioniamo Invia. A questo punto "magicamente" sull'altro cellulare apparirà la scritta "È arrivato un Messaggio" e sarà possibile visualizzarlo tramite l'opzione *Leggi* del menu.

CONCLUSIONI

L'applicazione che abbiamo descritto è un semplice esempio di utilizzo delle Bluetooth API su dispositivi J2ME. Una volta costruite le classi che si occupano della comunicazione con la radio Bluetooth, la nostra applicazione manipola dei semplici flussi di byte, per i quali abbiamo definito un protocollo. Poter manipolare direttamente da J2ME il chip Bluetooth del nostro device apre sicuramente tantissimi scenari di utilizzo.
La nostra stessa applicazione è completamente migliorabile. Infatti potrebbe essere salvato un vero e proprio profilo dell'utente, nel quale poter inserire anche delle risposte ad un questionario per poter far conoscere persone con profili simili. Grazie alle MMAPI potremmo scattare una foto da inserire nel profilo personalizzata o un video di presentazione.
Inoltre potremmo rendere possibile per l'utente la scelta delle persone a cui mandare il proprio profilo, cosa che adesso non avviene.

Federico Paparoni

L'AUTORE

Federico Paparoni è laureato in Ingegneria Informatica. È specializzato nello sviluppo di applicazioni client-server per terminali mobili e collabora con delle aziende di telefonia. I suoi interessi principali riguardano le piattaforme J2ME e J2EE.

ALTRI TOOLS PER IL BLUETOOTH

Oltre alla programmazione in J2ME ci sono altri modi per poter usufruire della connessione Bluetooth nello sviluppo di un'applicazione. La piattaforma di *SymbianOs*, sistema operativo presente sui più moderni smartphone, e WindowsMobile 2003 mettono a disposizione delle API per l'utilizzo del Bluetooth. Inoltre c'è la possibilità di sviluppo anche su pc desktop, grazie a varie implementazioni di Stack Bluetooth come quelle riportate qui di seguito.

AVETANABLUETOOTH
Un'implementazione bluetooth aderente alle specifiche JSR82. Può essere utilizzato sui principali sistemi operativi grazie a differenti Stack Bluetooth supportati (Widcomm per Win, Apple's System-Stack per Mac e BlueZ per Linux). http://www.avetana-gmbh.de/avetana-gmbh/produkte/jsr82.eng.xml

BLUECOVE
Un'altra implementazione opensource che per il momento supporta soltanto lo stack Bluetooth fornito dalla Microsoft con Windows XP Service Pack 2. http://sourceforge.net/projects/bluecove/

Grazie a queste implementazioni è possibile interfacciare il nostro programma Java con una chiave Bluetooth e quindi comunicare con altri dispositivi raggiungibili. Per capire meglio il protocollo su cui si basano diversi servizi Bluetooth da poter utilizzare, come ad esempio il classico trasferimento file, possiamo far riferimento a diversi siti:

http://www.palowireless.com/infotooth/knowbase.asp **La più grande raccolta di informazioni riguardanti il Bluetooth al momento attuale. È possibile capire dalle FAQ come sono strutturati i diversi servizi e protocolli utilizzati con il Bluetooth**

http://jabwt.com/ **Il sito di riferimento per informazioni riguardanti le JABWT (Java Api For Bluetooth Wireless Technology).**

http://www.benhui.net **Sito per sviluppatori J2ME, particolarmente famosa la sezione dedicata al Bluetooth dove è possibile trovare diverse guide per lo sviluppo di applicazioni**

Voice User Interface, verso il futuro

Applicazioni parlanti in Java

Utilizzeremo un sintetizzatore vocale: FreeTTS. Importeremo delle news prese da siti web e faremo in modo che vengano lette ad alta voce dal software

Nella moderna concezione delle caratteristiche che un'applicazione dovrebbe avere viene sempre più indicata la presenza di una *VUI (Voice User Interface)*. Rispetto ad un tradizionale programma che stampa a schermo le informazioni richieste, il poterle ascoltare tramite gli speaker del computer è sempre una feature molto apprezzata dagli utenti. Proprio per questo motivo si parla molto di tecnologie vocali, soprattutto embedded all'interno di dispositivi mobili come palmari e cellulari. Per lo sviluppatore Java esiste la specifica *JSAPI (Java Speech API)* che definisce delle API che consentono di includere tecnologie vocali all'interno delle proprie applicazioni. Utilizzando una delle implementazioni disponibili per JSAPI creeremo un semplice programma "parlante". Sfrutteremo delle news in formato *RSS (Really Simple Syndication)* prese da diversi siti per creare dei contenuti validi per la nostra applicazione (oltre al banale Hello World).

ARCHITETTURA DELL'APPLICAZIONE

L'applicazione è molto semplice nella sua struttura. Abbiamo prima di tutto una classe Java, *VoiceNewsGUI*, che permette all'utente di scegliere la tipologia di news sulla quale vuole essere sempre aggiornato. C'è poi la classe XMLController che viene richiamata ogni 10 minuti dal nostro programma per controllare se ci sono nuove notizie e, nel caso, notificarle all'utente tramite un'implementazione JSAPI. Queste informazioni vengono salvate in un semplice vettore che serializzeremo e deserializzeremo rispettivamente alla fine e all'avvio del programma per avere così salvate le news già ascoltate dall'utente. La parte "parlante" dell'applicazione sarà appunto contenuta all'interno della classe *VoiceShell*, che servirà a sintetizzare le stringhe che le vengono date in input.
Come già detto, l'applicazione risulta abbastanza

SELEZIONE DELLE NEWS

```
JradioButton r1=new JRadioButton("Salute");
JradioButton r2=new JRadioButton(
                    "Economia e Finanza");
…
…
r1.addActionListener(radiolistener);
r2.addActionListener(radiolistener);
…
….
getContentPane().add(r1);
getContentPane().add(r2);
```

1 Nella costruzione dell'interfaccia grafica utilizziamo dei pulsanti radio per selezionare la tipologia di news, utilizzando dei *JRadioButton*.

RICHIESTA FEED RSS

```
DOMParser parser = new DOMParser();
try {
parser.setFeature("http://xml.org/sax/
            features/validation", true);
}catch (Exception e){
  System.err.println (e);}
  try{
        parser.parse(xmlFile);
        Document document =
                 parser.getDocument();
       current=(Vector)nn.clone();
       leggi(document,0);
}catch (Exception e){
System.err.println (e);
}
```

2 In base alla richiesta che viene effettuata scarichiamo il file xml contenente i feed RSS, passandolo come argomento al parser DOM di Xerces.

AGGIORNAMENTO DEI CONTENUTI

```
for(int i=0;i<current.size();i++) {
  if (nn.contains((String
                 )current.elementAt(i)))
    nn.removeElement((String
                 )current.elementAt(i));
  else {
    nn.addElement((String
                 )current.elementAt(i));
  }
}
```

3 Nelle notizie che vengono scaricate sono presenti anche le vecchie notizie, quindi effettuarne un controllo su quelle già presenti nel nostro sistema.

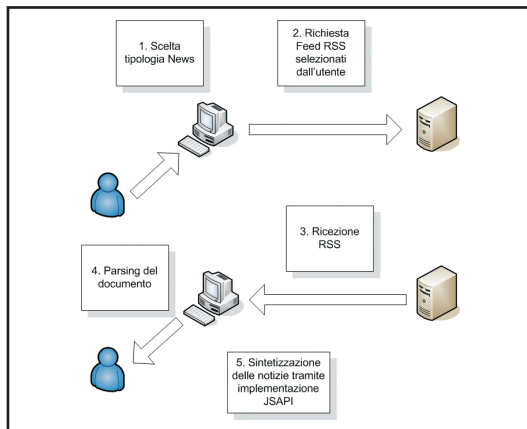



Fig. 1: ***Architettura dell'applicazione***

semplice. Le uniche difficoltà sono il parsing del file RSS, che è un file XML in cui vengono riportate le ultime news pubblicate su un sito, e il fatto di utilizzare un'implementazione di JSAPI che ci permetta di testare senza problemi la nostra applicazione.

JSAPI E FREETTS

JSAPI è una specifica sviluppata dalla SUN che definisce una serie di interfacce utilizzabili per implementare applicazioni vocali. I due aspetti fondamentali di questa specifica sono la sintesi e il riconoscimento vocale. Per meglio capire come è strutturato il mondo delle interfacce vocali dobbiamo dire che insieme a JSAPI sono stati definiti due linguaggi da utilizzare nella costruzione di un'interfaccia vocale, JSGF e JSML. *JSGF (Java Speech Grammar Format)* è un formato per la rappresentazione di grammatiche, le quali poi saranno utilizzate dal motore di riconoscimento vocale. Ecco un semplice esempio di grammatica in JSGF:

```
#JSGF V1.0;
// Definiamo il nome della grammatica
grammar SimpleCommands;
// Define the rules
public <Command> = [<Polite>] <Action> <Object>
                                    (e <Object>)*;
<Action> = apri | chiudi | abbassi;
<Object> = il browser | risorse del computer | la posta;
<Polite> = per piacere;
```

Con questa semplicissima grammatica vengono definite delle frasi del tipo "per piacere apri il browser e la posta". Le parentesi quadre stanno ad indicare che in quel comando può essere presente o non essere presente il token *Polite*. L'asterisco invece significa che possiamo trovare diverse occorrenze ripetute di Object nel nostro comando. Partendo da grammatiche semplici possiamo quindi creare piano piano delle grammatiche complicate. *JSML (Java Speech Markup Language)* è invece un linguaggio, derivato dall'XML, per definire la struttura delle interfacce vocali; questo è stato definito per minimizzare lo sforzo nel creare delle buone interfacce vocali e permettere di avere in mano uno strumento potente come un sintetizzatore vocale, senza entrare troppo nei dettagli riguardanti l'implementazione. Infatti grazie ad esso possiamo semplicemente definire la struttura, la pronuncia del testo. Il processo attraverso il quale viene utilizzato JSML, per creare

Fig. 2: ***Utilizzo di JSML nella sintesi vocale***

Java Speech API è una specifica della SUN, rilasciata il 26 Ottobre 1998. Attualmente non è contenuta nell'SDK perché non esiste una implementazione della SUN. Nelle FAQ di JSAPI è comunque presente una lista di prodotti che rispettano in parte la specifica 1.0
http://java.sun.com/products/java-media/speech/forDevelopers/jsapifaq.html#implementation

SCELTA DELLA VOCE

```
VoiceManager voiceManager =
              VoiceManager.getInstance();
Voice helloVoice =
       voiceManager.getVoice(voiceName);
```

4 **Ora che abbiamo le nuove notizie dobbiamo istanziare il motore di sintesi vocale. Per utilizzare l'implementazione JSAPI di FreeTTS dobbiamo prima di tutto inizializzare un VoiceManager, dal quale poi prendiamo la voce (Voice) che verrà utilizzata nel programma. La voce deve essere correttamente installata sul sistema.**

SINTESI VOCALE

```
if (helloVoice == null) {
         System.out.println("Impossibile
trovare la voce specificata");
      }
else {
helloVoice.allocate();
          helloVoice.speak(arg);
          helloVoice.deallocate();
}
```

5 **Inizializzata la voce passiamo semplicemente come argomento il testo della notizia che abbiamo estrapolato e sentiremo dagli speaker del computer se abbiamo fatto un buon lavoro.**

AVVIARE L'APPLICAZIONE

```
java –Dmbrola.base=
                /home/federico/mbrola –jar
                     bin/VoiceNews.jar it_1
```

6 **Per poter avviare l'applicazione che usa voci MBROLA e FreeTTS, dopo averli installati correttamente sul sistema, dobbiamo definire da riga di comando la variabile mbrola.base e metterla uguale alla directory dove abbiamo installato MBROLA.**

un documento che poi verrà letto dal sintetizzatore, è semplice. Ipotizziamo di avere dei dati da leggere in formato testuale. Questi vengono recuperati dalla nostra applicazione. A questo punto avviene la creazione del documento JSML, basandosi sul testo di informazione e su come vogliamo strutturare l'interfaccia vocale. In seguito questo documento viene dato in pasto ad un sintetizzatore vocale, il quale supporta il nostro linguaggio di markup ed, in base ad esso, emette il suono.

VOICEXML

VoiceXML è un linguaggio di markup per VUI (Voice User Interface) nato nel 1999.
È incluso all'interno del W3C Speech Interface Framework ed è attualmente uno dei modi più semplici per creare interfacce vocali.
Infatti esistono molte aziende su internet che offrono SDK per lo sviluppo di VUI.
Una delle più famose quella dell'IBM, che ha utilizzato VoiceXML per definire un ulteriore linguaggio per lo sviluppo di applicazioni multimodali, X+V
http://www.voicexml.org/
http://www.ibm.com/developerworks/library/wi-xvlanguage/
http://www.loquendocafe.com/

Questi due linguaggi fanno anche parte del W3C Speech Interface Framework, strumento pensato per gli sviluppatori di interfacce vocali. JSAPI rimane comunque solo una specifica SUN. L'implementazione, attualmente, più semplice e immediata da usare è FreeTTS, un sintetizzatore vocale scritto totalmente in Java. Adesso passiamo subito all'implementazione della nostra applicazione, approfondendo poi gli strumenti che utilizzeremo.

APPROFONDIMENTO

Il Voice Browser Working Group, istituito nel 1999 dal W3C, si occupa di linguaggi di markup per creare dei browser vocali. Il W3C Speech Interface Framework, introdotto dal VBWG, è un insieme di specifiche per la creazione di interfacce vocali.
http://www.w3.org/TR/voice-intro/

SERVIZIO UTILIZZATO

Per lo sviluppo della nostra applicazione ci appoggeremo ad un servizio di feed RSS disponibile su internet. Navigando su internet vi sarà capitato sicuramente di imbattervi in siti che utilizzano nella loro interfaccia grafica delle news provenienti da altri siti. Queste notizie vengono semplicemente importate grazie appunto agli RSS (Rich Site Summary o Really Simple Syndication), un file XML che dinamicamente viene aggiornato con le ultime notizie inserite in un sito web.
La struttura dell'RSS che andremo ad utilizzare è la seguente:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<rss version="2.0">
<channel>
  <title>Notizie sul Calcio </title>
   <link>http://www.sito.it/notiziecalcio.html</link>
   <description>Le ultime notizie sul calcio</description>
   <language>it</language>
   <copyright>http://www.sito.it</copyright>
 <item>
<title>Serie A non si gioca</title>
<link> http://www.sito.it /approfondimento1.html</link>
</item>
<item>
<title>Serie B si gioca</title>
<link> http://www.sito.it /approfondimento2.html</link>
</item>
</channel>
</rss>
```

Come possiamo vedere la struttura di questo file è abbastanza comprensibile. Ogni news viene incapsulata all'interno del tag item,dove troviamo il titolo e il link per l'approfondimento della notizia. Queste sono le sole informazioni che noi dobbiamo estrarre per notificarle all'utente e memorizzarle all'interno della nostra applicazione.
Passiamo ora alla vera e propria implementazione del programma.

IMPLEMENTAZIONE

Iniziamo parlando dell'interfaccia grafica, grazie alla quale il nostro utente potrà selezionare la tipologia di informazioni da utilizzare. Per semplificare la scrittura del codice, e per evitare soprattutto di far inserire all'utente dei siti che utilizzano dei feed RSS non ben strutturati, includeremo direttamente nel codice vari link, nei quali è possibile trovare degli RSS con varie informazioni che sappiamo già essere ben formate. Quindi nella costruzione della nostra interfaccia grazie a delle semplici JRadioButton sarà possibile per l'utente scegliere varie tipologie di news:

```
…
r1=new JRadioButton("Salute");
r2=new JRadioButton("Economia e Finanza");
r3=new JRadioButton("Societa");
r4=new JRadioButton("Scienza");
r5=new JRadioButton("Sport");
r6=new JRadioButton("Spettacolo e Cultura");
…
```

Dopo aver scelto, l'utente avvia lo scaricamento delle news ed entra in azione la classe *XMLController*, che periodicamente le controlla e aggiorna quelle già presenti sul computer. Questa classe viene implementata estendendo *TimerTask*, un timer apposta che richiama il metodo *run()* periodicamente (periodo che viene scelto nella costruzione della classe):

```
java.util.Timer t=new java.util.Timer();
XMLController xmlc=new XMLController(selezionato);
  t.schedule(xmlc,60000,60000);
```

In questo modo facciamo schedulare il nostro timer ogni 60 secondi, periodo di tempo che possiamo tranquillamente aumentare, perché è difficile che ci siano nuove notizie ogni minuto. A questo punto dobbiamo effettuare il parsing del file RSS. Per fare ciò utilizzeremo le librerie Xerces di Apache. Una volta passato al parser, il file remoto da controllare, implementeremo un metodo che lo scorrerà alla ricerca dei tag title e link, gli unici che saranno utilizzati da noi per indicizzare le news all'interno del programma:

```
parser.parse(xmlFile);
        Document document = parser.getDocument();
        leggiNews(document,0);
```

Il metodo *leggiNews* viene richiamato ricorsivamente per ogni livello di profondità dell'albero di parsing. Per ogni foglia dell'albero dovremo controllare la presenza dei tag e aggiungerli al vettore di notizie che risiede sul nostro computer, evitando di inserire quelle già presenti.

```
private void leggiNews(Node node, int numLevelsDeep) {
 String temp="";
 boolean vecchia=false;
 int type = node.getNodeType();
 if (type == Node.ELEMENT_NODE) {
    if("title".equals(node.getNodeName())) {
      temp=""+node.getFirstChild().getNodeValue();
      //System.out.print("\n"+node.getFirstChild(
                                  ).getNodeValue());
    }
    else if("link".equals(node.getNodeName())) {
       temp=""+node.getFirstChild().getNodeValue();
       //System.out.print("\n"+node.getFirstChild(
                                  ).getNodeValue());
    }
     for(int i=0;i<current.size();i++) {
        if (((String)current.elementAt(i)
                                  ).equals(temp)) {
           vecchia=true;
           current.removeElementAt(i);
        }
      }
       if (!vecchia)
          current.addElement(temp);  }
      NodeList children = node.getChildNodes();
       if (children != null) {
          for (int i=0; i<children.getLength(); i++) {
             leggi(children.item(i), numLevelsDeep+1);
          }
      }
   }
```

Ora abbiamo a disposizione le notizie ricercate all'interno di un vettore. A questo punto dobbiamo iniziare a utilizzare la classe che si occuperà di pronunciarle, ovvero *VoiceShell*. In questa classe utilizzeremo FreeTTS, il sintetizzatore di voce, che ci permette di far parlare la nostra applicazione, avendo semplicemente a disposizione la stringa che vogliamo sentire. I package necessari per la nostra semplice applicazione sono i seguenti:

```
import com.sun.speech.freetts.Voice;
import com.sun.speech.freetts.VoiceManager;
```

Con queste due righe importiamo le cose fondamentali di cui abbiamo bisogno, ovvero la classe *Voice* che rappresenta la voce che utilizzeremo e la classe *VoiceManager*, grazie alla quale possiamo scegliere la voce da utilizzare. Enumeriamo prima di tutto le voci presenti nella libreria e poi, dopo aver controllato se è presente quella che vogliamo utilizzare, possiamo allocarla e passarle come parametro la stringa che verrà sintetizzata:

```
VoiceManager voiceManager = VoiceManager.getInstance();
Voice[] voices = voiceManager.getVoices();
for (int i = 0; i < voices.length; i++) {
System.out.println("    " + voices[i].getName()+ " (
              " + voices[i].getDomain() + " dominio)");
}
String voiceName = (args.length > 0)? args[0]:
                                         "default_it";
Voice helloVoice = voiceManager.getVoice(voiceName);
if (helloVoice == null) {
   System.out.println("Errore: impossibile trovare la
                                 voce selezionata");
}
else {
   helloVoice.allocate();
   helloVoice.speak("News "+parametroPassato);
helloVoice.deallocate();
}
```

Abbiamo così realizzato la nostra classe "parlante", la quale viene richiamata da *XMLController* ogni

Fig. 3: **Componenti del sintetizzatore FreeTTS**

Per importare e creare delle voci per il nostro sintetizzatore FreeTTS possiamo utilizzare anche FestVox, un progetto simile a MBrola che ha come obiettivo quello di permettere di creare delle voci in maniera semplice, alla portata di tutti.
http://festvox.org/
http://freetts.sourceforge.net/tools/FestVoxToFreeTTS/README.html

Home page del progetto MBrola
http://tcts.fpms.ac.be/synthesis/mbrola.html

volta che viene notificato un aggiornamento sulle news. Questa classe può essere chiaramente inserita anche in un'altra applicazione per costruire delle interfacce vocali più strutturate e complesse di questo semplice approccio.

AVVIO E TEST

Prima di tutto dobbiamo installare sul nostro sistema FreeTTS, gratuitamente scaricabile dal seguente sito *http://sourceforge.net/project/showfiles.php?group_id=42080* oppure prelevandolo dal CD allegato ad ioProgrammo. FreeTTS è un progetto dello Speech Integration Group di SUN Microsystems Laboratories e può essere impiegato come implementazione JSAPI 1.0 ma anche come semplice TTS, remoto o locale, per le nostre applicazioni. È possibile anche importare FreeTTS in un applet, ma ci sono chiaramente i soliti problemi di sicurezza che riguardano Java.

Queste librerie richiedono almeno Java 1.4.0 installato sul sistema per poter funzionare. Sul sito sono disponibili tre diversi pacchetti, la distribuzione binaria, i sorgenti ed anche una versione, detta di test, che permette di testare l'applicazione che usa FreeTTS con JUnit. Una volta scaricata la distribuzione binaria dobbiamo settare l'ambiente JSAPI sul nostro sistema andando nella cartella in cui abbiamo decompresso FreeTTS, entrando in lib e avviando, a seconda che il nostro sistema operativo sia Linux o Windows, rispettivamente *jsapi.sh* o *jsapi.exe*.

Dal sito di FreeTTS è possibile scaricare numerosi esempi. In molti viene utilizzata la parte di FreeTTS che implementa le JSAPI.
http://freetts.sourceforge.net/docs/index.php

Fig. 4: ***Applicazione avviata***

Xerces è un parser XML, scritto in Java, distribuito gratuitamente da Apache XML Project. Questo package contiene un parser DOM e un parser SAX, permettendo così una semplici analisi di file XML.
http://xml.apache.org/xerces-j/

Questa fase però non è necessaria per la nostra applicazione, visto che non utilizziamo nessuna libreria standard JSAPI nel nostro piccolo esempio. All'interno di FreeTTS sono già presenti diverse voci sintetizzate, tutte in inglese. Per inserire delle voci italiane nel package di FreeTTS possiamo utilizzare tool come MBrola e FestVox. Sul sito della MBrola sono già presenti delle voci italiane che possiamo usare, limitando però l'applicazione ad essere eseguita soltanto sul sistema operativo Linux, perché per il momento non c'è il supporto a MBrola per Windows da parte di FreeTTS. La nostra applicazione, in italiano, sarà quindi distribuibile solo su Linux, per il momento. Lo scopo del progetto MBrola, sviluppato dai TCTS Lab, è quello di avere una vasta gamma di sintetizzatori vocali in tutte le lingue possibili. Una volta scaricato Mbrola e diverse voci italiane, installiamo il programma e decomprimiamo le voci all'interno della directory di installazione. Per poter poi utilizzare queste voci all'interno di FreeTTS dovremo specificare il path nel momento in cui lanciamo l'applicazione "parlante" nel seguente modo:

```
java -Dmbrola.base=/home/federico/mbrola
                    -jar bin/VoiceNews.jar it_1
```

Prima di fare ciò dobbiamo includere il *jar* di FreeTTS nel nostro classpath, ovvero includere *"freetts.jar"* che troviamo all'interno della cartella *lib*. In questo modo la nostra configurazione è completa e possiamo avviare l'applicazione.

Come mostrato nella **Figura 4**, ci troviamo davanti la scelta della tipologia delle news; una volta che abbiamo cliccato sul bottone *"Avvia"* il programma farà tutto da solo, avvertendoci di tanto in tanto della presenza di una nuova notizia.

CONCLUSIONI

JSAPI è una specifica ben fatta ma ancora lontana dall'essere implementata completamente per quanto riguarda FreeTTS, che è soltanto un sintetizzatore. Nonostante ciò è possibile realizzare con quest'ultimo degli interessanti programmi.

Nell'ambito di tecnologie vocali con supporto JSAPI dobbiamo per forza dare un'occhiata all'implementazione dell'IBM, Speech for Java. Questo prodotto, sviluppato su ViaVoice, permette di implementare un riconoscimento vocale oltre alla sintesi. Inoltre ha il supporto alla lingua italiana per Windows, però si appoggia a ViaVoice che deve essere installato sul computer dove si esegue il programma. Implementazioni a parte, il vocale ultimamente ha mosso il mercato tecnologico, infatti basta vedere come le tecnologie vocali siano cambiate totalmente a partire dal 1999 con l'introduzione del VoiceXML (linguaggio di markup per lo sviluppo di VUI) e dell'architettura che gli gira intorno. Il riconoscimento e la sintesi vocale è tuttora presente su diversi palmari e cellulari e rappresentano sicuramente degli strumenti che muovono la frontiera tecnologica. Avere a disposizione sempre più contenuti multimediali è sempre un punto a favore per le nostre applicazioni.

Federico Paparoni

Programmare ad oggetti anche i database relazionali

Persistenza dei Dati in PHP

In questo numero daremo largo spazio a Hibernate, etichettandolo come un tool per la persistenza dei dati orientato a Java. Ma cosa si intende per persistenza dei dati? Esiste qualcosa di analogo in PHP?

Utilizza questo spazio per le tue annotazioni

I programmatori PHP sono abituati a fare i conti con MySQL oppure con altre forme di database. Fino alla versione 4 di PHP; a causa di un supporto agli oggetti non ancora completo, l'approccio allo sviluppo era quasi completamente procedurale. Con la versione 5 di PHP le cose sono nettamente migliorate. L'approccio alla programmazione comincia a diventare ad oggetti, come nella maggior parte dei linguaggi che si rispetti. Programmare ad oggetti però cozza con la logica dei database che sono strutturati in modo relazionale e non hanno nessun legame con gli oggetti. Perciò nella migliore delle ipotesi, un approccio ad oggetti ad un database MySQL potrebbe prevedere qualcosa del genere:

```
$nome_utente='jaco';
$host = 'localhost';
$database='ioprogrammo      ';
$password='obfuscated';
$dsn = "mysql://$nome_utente:$password@$host/
                                        $database";
$db = DB::connect($dsn);
if (DB::isError($db)) {
      die ($db->getMessage());}
$sql='select * from redattori';
$result=$db->query($sql);
print "<pre>";
print_r($result);
print "<pre>";
$db->disconnect();
```

Si osserva facilmente che i campi del database non sono trattati come proprietà di un qualche oggetto. Viene utilizzato un oggetto della classe *Pear* per accedere al database, ma il valore restituto è pur sempre un array anche se contenuto in un oggetto di classe *db_result.* Il legame con la programmazione ad oggetti è estremamente debole. Se si eccettua il fatto che per accedere al db viene utilizzato un oggetto di classe *Pear,* non esiste in realtà alcun legame fra i campi che contengono il database e un'eventuale classe che li descrive. Dovremmo manualmente creare una classe, e mappare gli oggetti con i campi del database. Questo tipo di mapping fra oggetti e campi del database è la struttura su cui si fondano i tool di persistenza dei dati. Un tool di persistenza dei dati è un layer per creare una sinergia stretta tra oggetti e campi di un database.

PROPEL

Come tool di persistenza dei dati utilizzeremo per i nostri scopi: *"Propel".* Non è l'unico strumento del genere utilizzabile con PHP, tuttavia per semplicità d'uso, flessibilità, potenza, concetti utilizzati per la realizzazione del suo scopo, ci sembra uno dei più interessanti. Fra le altre cose è disponibile come Package PEAR. Possiamo considerare Propel come diviso in due parti:

1) Un generatore di classi e database
2) Un engine che fornisce una serie di classi base

L'utente dà in pasto al generatore di classi una serie di file XML e di configurazione. Sulla base delle indicazioni dell'utente, il generatore crea le classi necessarie ad utilizzare il tool di persistenza dei dati. Le classi generate sono per lo più derivate da quelle contenute dall'engine oppure sfruttano queste classi per creare oggetti e metodi propri.

STRUTTURA DI PROPEL

La struttura del file system di un computer al cui interno giri un web server attrezzato con PHP 5, po-

trebbe essere la seguente:

Notate la presenza della directory *'Propel'* nella root delle classi *PEAR* e la presenza di *Propel_generator* all'interno della directory *data*. Le due directory contengono rispettivamente l'*Engine* e il *Generator*. Allinterno della directory *propel_generator* è contenuta la sottodirectory *projects*. All'interno di *projects* in directory separate creeremo i file di configurazione necessari alla generazione delle classi di progetto e dello schema del database.

I FILE DI CONFIGURAZIONE

Il modo più semplice di esplicare i concetti che stanno alla base di Propel è quello di procedere con un esempio pratico. Supponiamo di avere ricevuto una commissione per la costruzione di una web application che consenta di registrare i comunicati stampa ricevuti da una redazione di una testata informatica. A ogni comunicato registrato dovrà essere associato ovviamente un numero di protocollo. Il Database che contiene l'archivio dei comunicati stampa, potrebbe essere formato dai seguenti campi.

```
[ID_COMUNICATO]
[NUMERO_PROTOCOLLO]
[BODY_COMUNICATO]
[AGENZIA_DI_COMUNICAZIONE]
[BRAND_COMUNICATO]
```

In un primo momento possiamo considerare ogni campo come una semplice riga di testo. In un momento successivo potremmo anche pensare, ad esempio ad *[AGENZIA_DI_COMUNICAZIONE]* come ad un record contenente informazioni sull'agenzia che ha mandato il comunicato. Allo stesso modo potremmo pensare che *[BRAND_COMUNICATO]* sia un record contenente informazioni sul Marchio oggetto del comunicato stampa. Ovviamente questi dati così strutturati dovrebbero essere messi in relazione fra loro. Proviamo a fare un'ipotesi di struttura XML per l'esempio proposto

```
<?xml version="1.0" encoding="ISO-8859-1"
                                standalone="no"?>
<!DOCTYPE database SYSTEM "../dtd/database.dtd">
<database name="comunicati_stampa"
                          defaultIdMethod="native">
 <table name="comunicato" description="Contiene i
             comunicati archiviati dalla segreteria">
   <column
    name="id_comunicato"
    required="true"
    primaryKey="true"
         autoIncrement="true"
    type="INTEGER"
    description="Id autoincrementale del comunicato
                                        stampa"/>
   <column name="numero_di_protocollo"
       required="true" type="VARCHAR" size="255"
    description="Numero di protocollo del comunicato"/>
   <column name="body_comunicato"
       required="true" type="VARCHAR" size="255"
                   description="Corpo del comunicato"/>
   <column name="id_agenzia" required="true"
      type="integer" description="Riferimento esterno
           alla tabella agenzie di comunicazione"/>
   <column name="id_brand" required="true"
    type="integer" description="Riferimento esterno
                        alla tabella dei brand"/>
<foreign-key foreignTable="agenzia_di_comunicazione">
    <reference local="id_agenzia"
                             foreign="id_agenzia"/>
   </foreign-key>
   <foreign-key foreignTable="brand_comunicato">
    <reference local="id_brand" foreign="id_brand"/>
   </foreign-key>
 </table>
```

ESTENDERE LE CLASSI PROPEL

L'architettura delle classi generate dal *Propel Generator* è tale da consentire di poterle estendere con facilità. Di fatto le classi generali ereditano da una classe *BaseXXX*. Ad esempio la classe *comunicato* eredita da una classe *BaseComunicato*. Pertanto è abbastanza semplice estendere la classe generale che è contenuta nel file *.PHP* nella radice del progetto e che ha lo stesso nome della classe.

PER INIZIARE

Prima di tutto dovete aver installato PHP e attivata l'estensione *XSL*. L'estensione è facilmente attivabile eliminando il commento alla linea *"extension=php_xsl.dll"* in *php.ini*. Se pensate di usare MySQL attivate anche l'estensione *"extension=php_mysql.dll"*. Il secondo passo è attivare le classi *PEAR* di base. Potete farlo facilmente avviando una console dos, portandovi nella directory di installazione di PHP ed eseguendo *"go-pear.bat"*. Una volta installate le classi *PEAR* di base potete installare *Propel-Generator* e *Propel-Engine.* Di solito da un prompt di msdos, portandosi all'interno della directory *PHP* i comandi necessari sono:

```
pear install http://creole.phpdb.org
                  /pear/creole-current.tgz
pear install http://propel.phpdb.org
            /pear/propel_runtime-current.tgz
pear install http://phing.info/pear
                      /phing-current.tgz
pear install http://propel.phpdb.org
           /pear/propel_generator-current.tgz
```

```
<table name="agenzia_di_comunicazione"
        description="Tabella delle agenzie">
  <column name="id_agenzia" required="true"
        primaryKey="true" autoIncrement="true"
        type="INTEGER"/>
  <column name="nome_agenzia" required="true"
        type="VARCHAR" size="128"/>
</table>
<table name="brand_comunicato" description=
        "Tabella dei marchi">
  <column name="id_brand" required="true"
        primaryKey="true" autoIncrement="true"
        type="INTEGER"/>
  <column name="nome_brand" required="true"
        type="VARCHAR" size="255"/>
</table>
</database>
```

PROPEL SUPPORTA IL CACHING DEI DATI?

No, a differenza di *Torque* da cui discende, non lo supporta.
È possibile aggiungere questa funzionalità, anche se in PHP le performance non aumenteranno di molto, estendendo il metodo *doSelect* nelle classi *Peer*.

QUALI VERSIONI DI MYSQL SONO SUPPORTATE?

La documentazione ufficiale riporta MySQL 4.0.x, noi abbiamo effettuato i nostri test con la versione 4.1 di MySQL senza incontrare grandi difficoltà.

Abbiamo volutamente semplificato, riducendo al minimo i campi. Notate però come ogni campo è descritto con una sintassi XML e che eventuali campi come ad esempio *[AGENZIA_DI_COMUNICAZIONE]* vengono ulteriormente splittati con dei sottocampi. Il formato XML si presta ottimamente a descrivere l'interazione che i vari campi devono avere fra loro.
Se avete copiato lo schema in questione, avete già compiuto circa un terzo del lavoro. Lo schema appena proposto rappresenta il primo dei file di configurazione necessari a Propel per eseguire i suoi compiti. Possiamo salvarlo nella directory *project/segreteria* con il nome *schema.sql* Gli altri due file di configurazione necessari prima di dare tutto in pasto a propel sono *build.properties* e *runtime-conf.xml*.
Conterranno rispettivamente, informazioni utili al *propel generator* per creare le classi e informazioni utili al *propel-engine* per utilizzare le classi durante la fase di esecuzione dell'applicazione. Il file *build.properties* potrebbe avere questo aspetto

```
propel.project = segreteria
propel.database = mysql
propel.database.url = mysql://localhost/segreteria
propel.targetPackage = segreteria
```

Il cui senso è intuitivo. Molto semplicemente viene dato un nome al progetto e si dicono poche altre cose rispetto al formato di database che verrà utilizzato. Il file *runtime-conf.xml* è più complesso

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<config>
  <log>
    <ident>propel-segreteria</ident>
    <level>7</level>
  </log>
    <propel>
      <datasources default="segreteria">
      <datasource id="segreteria">
          <adapter>mysql</adapter>
            <connection>
            <phptype>mysql</phptype>
            <hostspec>localhost</hostspec>
            <database>segreteria</database>
            <username>jaco</username>
            <password>obfuscated</password>
          </connection>
        </datasource>
      </datasources>
  </propel>
</config>
```

Anche in questo caso si tratta di un file sufficientemente esplicativo. Sostanzialmente vengono forniti i parametri che l'applicazione dovrà usare in fase di esecuzione, ad esempio le password e l'utente che verranno utilizzati per accedere al database.
Ambedue i file vanno salvati nella directory *project/segreteria* insieme a *schema.xml* che abbiamo gia prodotto in precedenza.

PROPEL GENERATOR

A questo punto siamo pronti per fare partire il *propel generator.* È sufficiente lanciare il prompt di msdos, portarsi nella directory *propel_generator* e lanciare

```
propel-gen projects\segreteria
```

A questo punto finita l'esecuzione del comando, ci rendiamo conto che è stata creata una nuova directory *"Build"* all'interno di *"Segreteria"* e che ha la seguente struttura:

La nostra attenzione al momento si concentrerà sulla directory sql, che contiene il file *schema.sql.*
Il file in questione contiene le istruzioni SQL per creare un database MySQL che rispetti le indicazioni contenute nel file *schema.xml* così per come lo avevamo concepito e che verrà utilizzato dal

Propel Engine per far funzionare l'applicazione. Tutto quello che dobbiamo fare è creare in mysql un database *"segreteria",* tale da potere essere usato dall'utente definito nel file *runtime-conf .xml.* L'ultimo passo è creare le tabelle del database in questione eseguendo i comandi sql contenuti nel file *schema.sql.*
Tutto è molto semplice:

```
mysql –ujaco –pobfuscated segreteria < schema.sql
```

PROPEL ENGINE

Siamo quasi pronti per programmare in PHP. Per potere utilizzare le classi generate dal *Propel Generator,* dobbiamo copiare i file che sono stati generati dalla directory *propel_generator/projects /segreteria/build/* alla directory *segreteria_app* all'interno della root del web server secondo la struttura che segue:

Il volume nell'unit… C _ Sistema
Numero di serie del volume: 809E-3FE8

PERCHÉ PROPEL USA CREOLE INVECE DELLE NORMALI CLASSI PEAR:DB O ADODB?

Perché si è ritenuto che *Creole* fosse un progetto più affidabile che implementasse in modo migliore l'astrazione dei dati. Inoltre essendo scritto da zero in PHP5 ne sfrutta tutti i vantaggi.

UN DATABASE DI CD MUSICALI

DEFINIAMO LO SCHEMA DEL DB

```
<?xml version="1.0" encoding=
            "ISO-8859-1" standalone="no"?>
<!DOCTYPE database SYSTEM
                        "../dtd/database.dtd">
<database name="musica"
                  defaultIdMethod="native">
  <table name="album">
    <column
      name="album_id"
      required="true"
      primaryKey="true"
            autoIncrement="true"
      type="INTEGER" />
    <column
      name="title"
      required="true"
      type="VARCHAR"
      size="255"
      description="Book Title"/>
    </table>
  </database>
```

1 **Creiamo il file XML che contiene lo schema del database e salviamolo con il nome *schema.sql* nella sottodirectory *projects* di *Propel Generator***

DEFINIAMO LE PROPRIETÀ IN FASE DI GENERAZIONE

```
propel.project = musica
propel.database = sqlite
```

2 **Creiamo il file che contiene le proprietà da usare in fase di generazione e salviamolo come *build.properties* nella sottodirectory *projects* di *Propel Generator***

DEFINIAMO LE PROPRIETÀ A RUNTIME

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<config>
  <log>
    <ident>propel-musica</ident>
    <level>7</level>
  </log>
  <propel>
    <datasources default="musica">
      <datasource id="musica">
        <connection>
          <phptype>sqlite</phptype>
          <hostspec>localhost</hostspec>
          <database>./test/musica.db
                              </database>
         <username></username>
         <password></password>
         </connection>
      </datasource>
      </datasources>
    </propel>
</config>
```

3 **Il file *runtime-conf.xml* contiene le informazioni che saranno usate dal progetto in fase di esecuzione, deve essere salvato nella sottodirectory *project* di *Propel Generator***

GENERIAMO E USIAMO LE CLASSI

```
propel-gen projects/musica
cd projects\musica
copy conf\* c:\htdocs\musica_app\conf\
cd build
cd classes
move musica c:\htdocs\musica_app\Classes
```

4 **A questo punto la struttura base per potere utilizzare le classi generate da *Propel Generator* è pronta. Non ci rimane che passare all'applicazione**

CREIAMO IL DATABASE

```
sqlite musica.db< schema.sql
copy musica.db c:\htdocs\musica_app\test
```

5 **Create il database utilizzando sqlite a linea di comando. Eliminate eventualmente dallo schema SQL i drop table che potrebbero generare errori**

INSERIAMO UN RECORD

```
<?
require_once 'propel/Propel.php';
Propel::init('./conf/musica-conf.php');
set_include_path("c:\cassiniweb
                  \musica_app\Classes;" .
                       get_include_path());
include_once 'musica/Album.php';
$album=new Album();
$album->setTitle('With Or Without You');
$album->save();
?>
```

6 **Notate che abbiamo usato il comando *set_include_path* per alterare il path di ricerca di PHP senza modificare il *php.ini.***
Il resto è abbastanza leggibile, abbiamo usato i metodi di propel per inserire un nuovo record. Sarà l'engine a convertire tutto in classiche funzioni SQL, il programmatore avrà una visione ad oggetti e non avrà percezione del meccanismo sottostante

Inoltre in *php.ini* sarà necessario settare i percorsi per potere utilizzare le classi.

IL PROGRAMMA PRINCIPALE

Siamo giunti al nocciolo della questione. Le operazioni preliminari possono essere sembrate tediose, ma il vantaggio che otteniamo in fase di programmazione dopo avere generato le classi base con il *propel_generator* è enorme. Proviamo a immaginare un primo file *index.php*

```
<?
function __autoload($class_name) {
   require_once 'segreteria/'.$class_name . '.php';}
require_once 'propel/Propel.php';
Propel::init('./conf/segreteria-conf.php');
$comunicato=new Comunicato();
$comunicato->setBodyComunicato('Questo è un
                                comunicato di prova');
$comunicato->save(); ?>
```

Come vedete abbiamo eseguito un ovverride di *__autoload($class_name)* per inserire le classi in modo automatico. Chiaramente le classi da inserire sono quelle generate da propel generator ed utilizzate adesso all'interno del nostro progetto. Viene poi inizializzata la connessione al database tramite una chiamata a *Propel::init* che prende come parametro il nome del file *segreteria-conf .php* generato da propel generator sulla base di *runtime-conf.xml*. Infine la parte più interessante è quella relativa all'uso dell'oggetto $*comunicato*. Notate che non c'è neanche una minima riga di SQL, stiamo lavorando completamente a oggetti. Il contenuto del database e la programmazione a oggetti hanno ora una relazione stretta!
Se provate a eseguire *index.php* scoprite però che viene generato un errore

```
[wrapped: Could not execute update [Native Error: Cannot
     add or update a child row: a foreign key constraint fails]
```

Di fatto *$comunicato* ha delle tabelle correlate e dei campi dichiarati come *"required"*, di cui non ci siamo minimamente interessati durante l'inserimento del dato. Proviamo a ripetere l'esperimento nel modo corretto

```
<?
function __autoload($class_name) {
   require_once 'segreteria/'.$class_name . '.php'; }
require_once 'propel/Propel.php';
Propel::init('./conf/segreteria-conf.php');
$agenzia = new AgenziaDiComunicazione();
$agenzia->setNomeAgenzia('Edmaster
Comunicazione');
$brand = new BrandComunicato();
$brand->setNomeBrand('Edizioni Master');
$comunicato=new Comunicato();
$comunicato->setNumeroDiProtocollo('0001');
$comunicato->setAgenziaDiComunicazione($agenzia);
$comunicato->setBrandComunicato($brand);
$comunicato->setBodyComunicato('Questo è un
                                comunicato di prova');
$comunicato->save(); ?>
```

Questo script funziona! Il dato viene inserito nelle varie tabelle. Non abbiamo scritto una riga si SQL. Propel engine si è curato di inserire i dati in modo tale che le relazioni fra le tabelle vengano rispettate. Notate anche che nella directory *"segreteria_app"* c'è ora un fantastico file di log *"propel.log"* che contiene appunto il log di tutte le operazioni effettuate sui record.

CERCARE UN RECORD

La ricerca di un record parte dalla definizione di un criterio di ricerca. Il criterio di ricerca più semplice è ovviamente basato sulla selezione della chiave:

```
$comunicato = new ComunicatoPeer();
print $comunicato->retrieveByPK(2)->
                                getBodyComunicato();
```

Utilizzando questa sintassi viene restituito l'unico oggetto la cui chiave primaria *ID_COMUNICATO* è uguale a 2. Per recuperare il contenuto del comunicato utilizziamo il metodo *getBodyComunicato*.
Non sempre però è utile selezionare l'unico oggetto corrispondente a un criterio. Più spesso ci trovermo a che fare con tabelle di oggetti restituite da un criterio, ad esempio:

```
$crit=new Criteria();
$crit->add(ComunicatoPeer::BODY_COMUNICATO,
                          "%prova%", Criteria::LIKE);
$results = ComunicatoPeer::
           doSelectJoinAgenziaDiComunicazione($crit);
foreach($results as $comunicato) {
   print $comunicato->getAgenziaDiComunicazione()->
         getNomeAgenzia()."<br>".$comunicato->
                   getBodyComunicato(). "\n"; }
```

Notate che in questo caso abbiamo definito un *Criterio* utilizzando la classe *Criteria*. La sintassi utilizzata è tale che vengono restituiti tutti i comunicati tali che all'interno del campo *"BODY_COMUNICATO"* sia contenuta la parola *"prova"*. Stampare a video la tabella corrispondente all'insieme degli oggetti restituiti è abbastanza semplice. È sufficiente utilizzare un ciclo di *For* recuperando i dati con la

APPROFONDIMENTI

QUALI SONO LE DIFFERENZE FRA PROPEL E TORQUE?

Propel **è un port di Torque.** ***Torque*** **è un progetto di Apache Foundation per la persistenza dei dati scritto in Java per applicazioni Java. Le differenze sono in molti casi marginali.** ***Propel*** **ad esempio supporta il** ***"cascading deletes"*** **che consente di eliminare oggetti che hanno una loro integrità referenziale. Altre differenze consistono per esempio nel metodo** ***Hydrate*** **che consente di non dover inserire il risultato di una query interamente in un array, ma poterlo splittare come segue:**

```
$rs=BookPeer::doSelectRS(
              new Criteria());
while($rs->next()) {
  $book = new Book();
  $book->hydrate($rs);
  // read $book values,
         save $book, etc.
  ...
}
```

classica sintassi ad oggetti.

CRITERI MULTIPLI

È ovviamente possibile utilizzare criteri complessi generati da due o più condizioni unite fra loro da un operatore logico. Ad esempio:

```
$crit=new Criteria();
$criterion = $crit->getNewCriterion(ComunicatoPeer::
        BODY_COMUNICATO, "%prova%", Criteria::LIKE);
$criterion->addAnd($crit->getNewCriterion(ComunicatoPeer::
        ID_COMUNICATO, "1", Criteria::GREATER_THAN));
$crit->add($criterion);
$results = ComunicatoPeer::
        doSelectJoinAgenziaDiComunicazione($crit);
foreach($results as $comunicato) {
  print $comunicato->getAgenziaDiComunicazione()->
        getNomeAgenzia()."<br>".$comunicato->
                getBodyComunicato(). "\n";}
```

La sintassi è leggermente più complessa, ma molto potente. Notate come ci siamo serviti di un nuovo oggetto *"$criterion"* che abbiamo utilizzato per creare un criterio complesso. I due criteri sono stati uniti tramite il metodo *"addAnd"* che corrisponde all'*AND* logico, poteva anche essere usato il metodo *"addOr"* che corrisponde all'*OR* booleano. È interessante anche notare l'operatore booleano *"GREATER_THAN"* che corrisponde al *"Maggiore di"* booleano. Il tipo di ricerca che abbiamo effettuato non fa altro che restituire una tabella di oggetti tali che il campo *"BODY_COMUNICATO"* contenga la parola "prova" e l'*ID_COMUNICATO* sia maggiore di 1.

CANCELLARE O MODIFICARE UN RECORD

Una volta compreso come vengono gestiti i criteri, la modifica o la cancellazione di un record diventa una mera applicazione di metodi. Ad esempio:

```
$crit=new Criteria();
$crit->add(ComunicatoPeer::ID_COMUNICATO,2);
$results = ComunicatoPeer::doSelect($crit);
foreach ($results as $comunicato) {
        $comunicato->setBodyComunicato($comuni-
cato->getBodyComunicato()."-Modificato");
        $comunicato->save();
}
```

Questo tipo di approccio modifica tutti i comunicati stampa il cui *ID_COMUNICATO* è due aggiungendo la parola *"Modificato"* al termine del body del comunicato. In realtà questo è un approccio puramente didattico. Va da se che esiste un solo comunicato il cui ID sia uguale a *"2"*. Un approccio del genere è utile quando il set di oggetti restituiti è una tabella contenente più di un record. Viceversa è più utile utilizzare una sintassi di questo tipo:

```
$comunicato = ComunicatoPeer::retrieveByPK(2);
$comunicato->setBodyComunicato($comunicato->
                getBodyComunicato() . "-modificato");
$comunicato->save();
```

Allo stesso modo è facile intuire che l'eliminazione di uno o più record è affidata al metodo doDelete

```
$crit = new Criteria();
$crit->add(ComunicatoPeer::BODY_COMUNICATO,
                        "%prova%",CRITERIA::LIKE);
ComunicatoPeer::doDelete($crit);
```

Elimina tutti i comunicati che contengono la parola prova.

CONCLUSIONI

Propel rappresenta un'ottima soluzione per gestire la persistenza dei dati. Il problema principale è entrare in sintonia con la sintassi e la logica del framework, una volta sostituita l'abitudine alla classica sintassi MySQL con quella legata agli oggetti proposti da Propel si noterà come la rapidità di sviluppo aumenterà in modo vertiginoso.

USARE SQLITE

Il tool per utilizzare I database a linea di comando può essere scaricato da http://www.sqlite.org/download.html, oppure lo trovate nella directory SQLite del CD allegato. Installarlo è semplicissimo, è sufficiente scompattare il file *.zip* in una directory appropriata e possibilmente aggiornare il percorso di ricerca del path di windows per potere utilizzare il comanda da qualunque directory.

OPERATORI BOOLEANI

Più di una volta abbiamo fatto riferimento all'uso degli operatori booleani, in special modo nella definizione dei criteri. Ma quali sono quelli utilizzabili? In realtà Propel riproduce in modo quasi identico il modello proposto da Torque - http://db.apache.org/torque/index.html - il layer per la persistenza dei dati proposto in Apache. Pertanto gli operatori booleani utilizzabili sono identici a quelli definiti in Torque e in particolare

ALT_NOT_EQUAL	Corrisponde a *"Diverso da"*
DISTINCT	Corrisponde al *DISTINCT* di MySQL, ritorna cioè righe di una colonna una sola volta anche se ne esistono duplicati
EQUAL	Corrisponde a *"Uguale a"*
GREATER_EQUAL	Corrisponde a *"Uguale o Maggiore di"*
GREATER_THAN	Corrisponde a *"Maggiore di"*
IN	Verifica se un dato è appartenente a un insieme
LESS_EQUAL	Corrisponde a *"Uguale o Minore di"*
LESS_THAN	Corrisponde a *"Minore di"*
LIKE	Si comporta esattamente come l'operatore *LIKE* di MySQL,
NOT_EQUAL	Corrisponde a *"maggiore o minore di"*
NOT_IN	Verifica la non corrispondenza ad un insieme

Propel è *CASE SENSITIVE* nell'applicazione dei criteri. È possibile effettuare ricerche *CASE INSENSITIVE* utilizzando una sintassi del tipo:

```
$c = new Criteria();
$c->add(AuthorPeer::FIRST_NAME, "max");
$c->setIgnoreCase(true);
```

Protocolli POP3 e SMTP: come usarli con .NET

Un client di posta in C#

Tutte le funzionalità che ci servono per scaricare i messaggi di posta, filtrarli e, eventualmente, farci segnalare i nuovi spam

Utilizza questo spazio per le tue annotazioni

Nel precedente articolo abbiamo realizzato un Server Antispam. Tale server aveva lo scopo di memorizzare gli indirizzi e-mail segnalati dagli utenti registrati al servizio e di renderli disponibili in formato XML. Ora non ci resta che realizzare il nostro client capace, da un lato di scaricare la posta via POP3, dall'altro di scaricare ed utilizzare le definizioni di Spam. L'aspetto che daremo al nostro client è quello di **Figura 1**. Per il download delle e-mail, ci serviremo del protocollo POP 3 e vedremo come usarlo in modo corretto in C#. Mettiamoci subito all'opera!

POP3 E C#

Basandosi sostanzialmente su una connessione di tipo TCP. L'utilizzo di POP3 in un'applicazione client presuppone la realizzazione di un Socket. Un socket è sostanzialmente un canale di comunicazione che consente l'invio e la ricezione di dati sfruttando un protocollo, nel nostro caso TCP.
Nel riquadro in basso è possibile seguire tutta la sequenza necessaria per implementare il protocollo POP3 in una applicazione C#

ANALIZZIAMO IL CLIENT

Dopo aver visto il protocollo POP3 e come questo si usa in C#, analizziamo più in dettaglio il funzionamento del client. Il primo aspetto che analizzeremo è il download dei messaggi.
Nel nostro client è implementata una classe per la gestione del protocollo POP3 (*pop3.cs*). In essa ci sono tutti i metodi che si occupano della creazione del canale di comunicazione, dell'invio dei comandi al server e della lettura dei risultati. Tali risultati dovranno poi essere gestiti in modo opportuno dal client. Gran parte della logica del software è implementata nel form principale (*form1.cs*) la cui interfaccia utente è visibile in **Figura 1**.

Fig. 1: Il client di posta

Il metodo che permette di scaricare la posta sul client è *DownloadMails()*. In esso viene istanziata la classe POP3

```
private void DownloadMails(){
  POP3 pop3 = new POP3();
  pop3.User = uInfo.mail_UserName;
  pop3.Pwd = uInfo.mail_Password;
  pop3.Pop = uInfo.mail_Server;
  rtbLog.AppendText(pop3.Connect()+"\r\n");
  pop3.USER();
  pop3.PASS();
```

viene effettuata la connessione al server e vengono inviate le credenziali. Se la connessione ha esito positivo (il server è raggiungibile e le credenziali inviate sono valide), possiamo inviare i comandi elencati in **Tabella 1.**
Quello che ci interessa fare è scaricare i messaggi sull'Hard Disk del nostro PC in modo da poterli leggere anche in modalità disconnessa. Dovendoli scaricare tutti, per prima cosa dobbiamo conoscere quanti messaggi ci sono. Facendo riferimento alla

Comando	Descrizione
USER nomeutente	Invio del nome utente
PASS password	Invio della password
STAT	Stato della casella di posta (numero di messaggi e dimensione degli stessi)
LIST	Elenco dei messaggi contenuti nella casella di posta
TOP n	Intestazione del messaggio n
RETR n	Contenuto del messaggio n
DELE n	Cancellazione del messaggio n
RSET	Reset della comunicazione
QUIT	Chiusura della sessione
NOOP	Non esegue nessun comando. Serve ad impedire che la connessione cada dopo il timeout del server

Tabella 1: elenco dei comandi supportati dal protocollo POP3

Tabella 1, vediamo che il comando da inviare è *STAT.* La risposta a tale comando, qualora ci fossero dei messaggi, sarà ad esempio:

```
+OK 3 5678
```

in cui *+OK* indica che il comando inviato è corretto e non ha causato errori, 3 indica il numero dei messaggi presenti e 5678 rappresenta la dimensione totale dei messaggi. Per facilitare l'estrazione del numero dei messaggi dalla risposta al comando *STAT,* nella classe POP3 è stata implementata una proprietà che ritorna direttamente il numero dei messaggi *(MailNumber)* che fa riferimento al metodo *ParseSTAT():*

```
int totmail = pop3.MailNumber;
```

Ora che ne conosciamo il numero, dobbiamo inviare il comando *RETR* per ogni messaggio presente sul server:

```
for(int i=1;i<=totmail;i++) {
  //invio del comando RETR e gestione della risposta
}
```

Prima però, dobbiamo considerare un paio di cose: quello che riceviamo dal comando *RETR* è una stringa contenente l'intero messaggio (header compresi). Da questa stringa dobbiamo estrarre le informazioni che ci permettono di rendere più presentabile il messaggio nonché di memorizzato sull'Hard Disk. Il primo problema si risolve creando un piccolo parser a cui passare l'intera stringa ricevuta dal server di posta. Il risultato del parser sarà quello di memorizzare in una serie di variabili i campi *FROM* (il mittente del messaggio), *TO* (il destinatario), *SUBJECT* (oggetto del messaggio), *BODY* (corpo del messaggio), *DATE* (data del messaggio) e *MESSAGEID* (ID del messaggio). Il funzionamento dettagliato del parser non verrà trattato in questo articolo in quanto si tratta semplicemente di una lunga serie di operazioni abbastanza ripetitive applicate ad una stringa (il nostro messaggio). Tutto il codice del parser è comunque contenuto nel file *ParseEmailMessage.cs.* Il secondo problema, relativo alla presentazione e memorizzazione dei dati, è risolvibile in vari modi. Il più immediato è quello di ricorrere ad un semplice Data Base per memorizzare i dati estratti dal parser. Sebbene questa sia la strada più imme-

Fig. 2: Realizzazione dell'interfaccia utente in Visual Studio

NOTA

Il testo dalla specifica RFC 1939 che descrive il protocollo POP3 può essere consultato a questo indirizzo: http://www.ietf.org/rfc/rfc1939.txt

IL PROTOCOLLO POP3

Prima di iniziare con la realizzazione del client, diamo uno sguardo al protocollo *POP3 (Post Office Protocol 3)*, il più utilizzato per il download della posta elettronica.
Tale protocollo è descritto dalla specifica RFC 1939 che fornisce le funzioni di base per leggere, scaricare e cancellare la posta da un Mail Server.
Un server che supporta il protocollo POP3 resta in ascolto sulla porta 110 attendendo che un client stabilisca su essa una connessione di tipo TCP. Una volta stabilita la connessione, il client potrà inviare una serie di comandi supportati e leggere le risposte del server.
Lo stato della connessione stabilita può assumere 3 stati:

- **AUTHORIZATION: è il primo passo. Una volta "chiamato" il server sulla porta 110, il client dovrà fornire delle credenziali valide per poter gestire la posta.**
- **TRANSACTION: è lo stato successivo all'autenticazione, qualora questa abbia avuto esito positivo. È solo in questo stato che è possibile inviare i comandi al server.**
- **UPDATE: il server POP3 passa in questo stato dopo il comando QUIT. In questa fase vengono eseguiti gli eventuali comandi di cancellazione precedentemente inviati e, successivamente, viene terminata la connessione TCP.**

Una tipica sessione POP3 avviene seguendo i seguenti passi:

1. **il client chiama il server POP3 sulla porta 110.**
2. **il server risponde con un messaggio di benvenuto e si porta nello stato di AUTHENTICATION.**
3. **il client invia la sequenza di UserName e, successivamente di Password. Se le credenziali sono valide, il server si porta nello stato di TRANSACTION. A questo punto sarà possibile inviare i comandi.**
4. **ad ogni comando inviato il server risponderà con un +OK seguito da eventuali dati qualora il comando inviato è corretto. –ERR invece se il comando inviato è errato.**
5. **la sessione viene terminata con il comando QUIT.**

L'elenco dei comandi inviabili al server è visibile in Tabella 1. Questo è quanto ci serve per scaricare la posta dal server. Ora non ci resta che implementarlo nel nostro client!

NOTA

Le classi socket sono indispensabili qualora si voglia realizzare un applicativo che debba aprire un canale di comunicazione. Per approfondire l'argomento relativo ai socket, un buon punto di partenza è http://msdn.microsoft.com/library/ita/default.asp?url=/library/ITA/cpref/html/frlrfsystemnetsocketssocketclasstopic.asp

diata (e molto utilizzata), vista la semplicità delle informazioni da memorizzare possiamo usare un *DataSet* tipizzato. Il *DataSet* altro non è che una rappresentazione dei dati residente in memoria. Sebbene generalmente un *DataSet* viene utilizzato in accoppiata ad un Data Base per la gestione dei dati in modalità disconnessa, nulla ci impedisce di usarlo come entità a sé stante. Per aggiungere un *DataSet* al nostro progetto è sufficiente cliccare con il tasto destro del mouse sull'icona relativa al progetto (in Visual Studio) e selezionare la voce *"Aggiungi nuovo elemento"*. Nell'elenco degli elementi che possono essere aggiunti, selezioniamo il *Dataset.xsd*, chiamiamolo *dsMail* ed aggiungiamolo al progetto.
Visual Studio ci offre un comodo strumento visuale per realizzare i nostri *DataSet*. È sufficiente aprire il *dsMail.xsd* appena aggiunto e dalla casella degli strumenti aggiungere gli elementi che ci interessano (**Figura 3**). Il primo sarà un *Element* che chiameremo Mail. Rappresenta in sostanza la nostra tabella. Come se stessimo operando su un Data Base, aggiungiamo all'elemento appena inserito i vari campi che ci interessano come in **Figura 3**. Ora che abbiamo il nostro parser ed il nostro "contenitore" per i messaggi scaricati, riprendiamo il codice del ciclo *for* e "trattiamoli". Per prima cosa creiamo una nuova istanza del nostro Data Set:

```
dsMail dsmail = new dsMail();
```

All'interno del ciclo *for* poi, passiamo al parser il messaggio ricevuto dal comando *RETR:*

Fig. 3: **Editing del DataSet dsMail in Visual Studio**

USIAMO IL PROTOCOLLO POP3

```
public POP3(string popServer, string
                userName, string password) {
   this.State = ConnectState.Disc;
   this.Port = 110;
   this.Pop = popServer;
   this.User = userName;
   this.Pwd = password;
}
```

1 **Attraverso il costruttore della classe POP3 impostiamo direttamente le proprietà relative al server da contattare (popServer) ed alle credenziali di accesso da inviare.**

```
public string Connect() {
   this.Server = new TcpClient(this.Pop,
                              this.Port);
   this.NetStream = this.Server.GetStream();
   this.RdStream = new StreamReader(
                    this.Server.GetStream());
   this.State = ConnectState.Authorization ;
   return(this.RdStream.ReadLine());
}
```

2 **Stabiliamo una connessione al server POP3 attraverso un socket TCP (riga 2) leggendone lo stream di risposta. Se inizia con +OK, la connessione è stata stabilita.**

```
private void SendCommand(string command)
{
   this.Data= command + CRLF;
   this.SzData = System.Text.Encoding
   .ASCII.GetBytes(this.Data.ToCharArray());
   this.NetStream.Write(
          this.SzData,0,this.SzData.Length);
}
```

3 **L'invio dei comandi al server POP3 è una operazione ripetitiva quindi ne incapsuliamo la logica in un metodo apposito: SendCommand**

```
public string RETR (int msgNumber) {
   string temp = "";
   if (this.State !=
              ConnectState.Transaction ) {
      temp = "La connessione non è nello
                    stato di Transazione";
   } else {
   this.SendCommand ("RETR "+
              msgNumber.ToString ());
    temp = this.ReadMultiLineResponse();}
  return(temp);}
```

4 **Un comando tipo è quello relativo alla ricezione del messaggio: *RETR n.* Se siamo nello stato *TRANSACTION*, il comando può essere inviato e possiamo leggerne la risposta.**

```
public string QUIT() {
   string temp;
   if (this.State != ConnectState.Disc) {
      this.SendCommand ("QUIT");
      temp = this.ReadSingleLineResponse();
      temp += CRLF + this.Disconnect();}
   else {
      temp = "Non connesso.";}
   return(temp);
}
```

5 **Inviando il comando *QUIT*, la sessione con il server POP3 verrà terminata. Internamente viene richiamato il *Disconnect()* che chiude il Socket.**

```
public string LeggiMail (int MailNumber){
POP3 pop3 = new POP3("Server",
             "NomeUtente", "Password");
pop3.USER();
pop3.PASS();
pop3.RETR(MailNumber)
pop3.Quit()
}
```

6 **Un esempio di richiesta tipo è questa: viene istanziata la classe POP3 che segue i passi spiegati nel relativo paragrafo dell'articolo.**

```
for(int i=1;i<=totmail;i++) {
  ParseEmailMessage parser = new ParseEmailMessage();
  parser.BeginParse(pop3.RETR(i));
```

creiamo una nuova *DataRow* (una nuova riga nel nostro DataSet), valorizziamone i campi con il risultato del parser e, per finire, aggiungiamola al DataSet:

```
DataTable dt = dsmail.Tables[0];
  DataRow dr = dt.NewRow();
  dr["From"] = parser.Get_MailFrom();
  dr["TO"] = parser.Get_MailTo();
  dr["CC"] = parser.Get_MailCC();
  dr["Subject"] = SpamFilter.Filter(CleanMail(
          parser.Get_MailFrom()), parser.Get_MailSubject());
  dr["Body"] = parser.Get_MailBody();
  dr["MailDate"] = parser.Get_MailDate();
  dr["MailID"] = parser.Get_MessageID();
  dsmail.Tables[0].Rows.Add(dr);
}
```

Dato che il ciclo verrà ripetuto per tutti i messaggi presenti, ogni riga del DataSet conterrà un singolo messaggio scaricato dal server.

USO DELLE DEFINIZIONI

Nel precedente articolo abbiamo visto come far generare al Server AntiSpam il file XML contenente le definizioni. Ora dobbiamo implementare nel client le funzionalità che ci consentono di ricevere il file delle definizioni e di usarlo. Il primo passo sarà quello di prendere la classe proxy relativa alla gestione del server antispam già analizzata nel precedente articolo (*AntiSpamManagerServices.cs*) e includerla nel nostro progetto. Tale classe infatti contiene il metodo *GetSpamList()* che ritorna l'elenco delle definizioni. Dato che il download delle definizioni richiede un accesso sicuro al Server AntiSpam, includiamo nel progetto anche la classe per la gestione del token di autorizzazione *AuthManager.cs* (queste due classi, compresa la teoria sull'accesso sicuro ai servizi web, sono state già trattate nell'articolo precedente quindi non verranno ripetute anche in questo). Nel nostro client, il metodo che si occupa di ricevere la lista delle definizioni e di salvarla sull'Hard Disk si chiama *DownloadSpamList* ed è così strutturato:

```
private void DownloadSpamList(){
  CreateToken(uInfo.ASService_UserName,
                              uInfo.ASService_Password);
  try{
    SoapEnvelope envelope = new SoapEnvelope();
    envelope.Context.Security.Tokens.Add(
                                    AuthManager.token);
    envelope.Context.Addressing.Action = new Action(
                                    "urn:GetSpamList");
    SpamManager sManager = new SpamManager();
    XmlDocument doc = new XmlDocument();
    doc.LoadXml("<?xml version=\"1.0\" encoding=\"utf-8
       \" ?><SpamDefinition>" + sManager.GetSpamList(
              envelope) + "</SpamDefinition>");
    doc.Save(Environment.CurrentDirectory +
                              @"\Definition.xml");
    rtbLog.AppendText("File delle definizioni salvato in " +
                         Environment.CurrentDirectory +
                              @"\Definition.xml"+"\r\n");
  }catch (Exception ex){
    MessageBox.Show("ERRORE: " + ex.Message);
  }
}
```

NOTA

Il Garbage Collector si occupa di gestire l'allocazione di memoria delle applicazioni .NET. Sinteticamente il collector libera la memoria non più necessaria e deframmenta quella allocata in modo da aumentarne le performance. Un buon punto di partenza per comprendere il collector è:
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dndotnet/html/dotnetgcbasics.asp

CREIAMO IL PROGETTO

1 **In Visual Sudio, scegliamo la voce *File/Nuovo/Progetto* e, dalla maschera *"Nuovo Progetto"* selezioniamo *"Progetti Visual C#"* come tipologia e *"Applicazione per Windows"* come modello. Dopo averne deciso nome e directory, confermiamo com *OK*.**

2 **Nel form principale dobbiamo inserire tutti gli elementi che compongono l'interfaccia utente del nostro client. Per la barra di navigazione di sinistra è stata scelta una *ListView*. Nella parte centrale alta posizioneremo un *DataGrid* mentre in quella bassa un Panel.**

3 **Il nostro client, oltre che dal form principale, sarà composto da altri elementi tra cui il DataSet tipizzato per la memorizzazione dei messaggi, la classe POP3, le classi per l'utilizzo di WSE ecc. Tutti gli elementi sono raggruppati sotto un unico progetto.**

La serializzazione è quel meccanismo che ci consente di trascrivere un oggetto in uno stream (su disco o memoria). Ci sono vari tipi di serializzazione (Binaria, Xml ecc), ognuna adatta a specifici contesti. Per chiarirsi le idee è possibile partire dal namespace *System .Runtime.Serialization*: http://msdn.microsoft.com/library/ita/default.asp?url=/library/ITA/cpref/html/frlrfsystemruntimeserialization.asp

Per comprendere meglio cosa è un DataSet, come utilizzarlo e come gestirlo, un buon punto di partenza è http://msdn.microsoft.com/library/ITA/cpguide/html/cpconCreatingUsingDataSets.asp?frame=true

Per la lettura del file XML con le definizioni, utilizzeremo l'oggetto *XmlTextReader*. Questo oggetto consente un accesso veloce ai dati contenuti in un file XML. Per approfondimenti consultare:

http://msdn.microsoft.com/library/ita/default.asp?url=/library/ITA/cpref/html/frlrfsystemxmlxmltextreaderclasstopic.asp **e** http://msdn.microsoft.com/library/ita/default.asp?url=/library/ITA/cpguide/html/cpconreadingxmldatawithxmltextreader.asp

Sostanzialmente, dopo aver creato il token con le credenziali dell'utente, costruiamo il messaggio SOAP da inviare al Server AntiSpam con la richiesta del file con le definizioni. Il metodo *GetSpamList()* della classe *proxy* ritorna una stringa che trasformeremo in XML con *XmlDocument.LoadXml()* e ne salveremo il risultato sull'Hard Disk. Ma ora che abbiamo il file con le definizioni, come lo usiamo? Abbiamo detto che il client dovrà "filtrare" le mail ricevute e anteporre a quelle che ritiene Spam una dicitura che le distingua dalle altre mail. Se riprendiamo il precedente paragrafo (quello relativo al download della posta), noteremo una istruzione particolare all'interno del ciclo *for*:

```
dr["Subject"] = SpamFilter.Filter(CleanMail(
        parser.Get_MailFrom()), parser.Get_MailSubject());
```

In particolare, a differenza degli altri campi, il Subject della mail viene filtrato prima di essere inserito nel DataSet di appoggio. Il metodo statico *SpamFilter.Filter()* si occupa proprio di valutare le mail in arrivo basandosi sul file XML contenete le definizioni. Vediamo quindi come è strutturato questo metodo:

```
public static string Filter(string mailFrom, string mailSubject){
  string NewMailSubject = string.Empty;
  XmlTextReader reader1 = new XmlTextReader(
     Environment.CurrentDirectory + @"\Definition.xml");
  while (reader1.Read()) {
    if ((reader1.NodeType == XmlNodeType.Element) &&
                              (reader1.Name == "Email")) {
      if (reader1.ReadString().ToLower() ==
                                  mailFrom.ToLower()){
        reader1.Read();
        if (reader1.IsEmptyElement) reader1.Read();
          reader1.ReadString();
          string rating = reader1.ReadString();
          reader1.Close();
        return "[***SPAM*** " + rating +" ] " +
                                  mailSubject; } }  }
  reader1.Close();
  return mailSubject;
}
```

Il primo step è quello di "leggere" il file XML delle definizioni. L'oggetto *XmlTextReader* (vedi box laterale) ci da un rapido accesso al contenuto di tale file. L'applicazione del filtro consiste semplicemente nel valutare se il mittente della mail (*mailFrom*) è contenuto nell'elenco delle definizioni. Qualora il riscontro dovesse avere esito positivo, la funzione passa alla lettura del *rating* ed antepone al *Subject* originale la dicitura:

```
return "[***SPAM*** " + rating +" ] " + mailSubject;
```

In questo modo, tutte le mail ritenute Spam verranno contrassegnate e saranno facilmente distinguibili da quelle che non lo sono.

SEGNALIAMO LO SPAM

Il nostro client è quasi completo. Possiamo scaricare la posta e salvarla sull'Hard Disk, possiamo scaricare ed usare il file con le definizioni di Spam ma non possiamo ancora segnalare nuovo Spam. Se riprendiamo la classe proxy *AntiSpamManagerServices.cs* noteremo che essa contiene un metodo denominato *SegnalaMail()* che accetta come input una stringa (l'indirizzo da segnalare). Questo metodo richiamerà il corrispettivo del Server AntiSpam che si occuperà dell'inserimento dell'indirizzo nel Data Base (lo abbiamo già visto nel precedente articolo). Ma come recuperiamo l'indirizzo da segnalare? È molto semplice: dal DataSet che abbiamo popolato con i dati delle mail ricevute! Abilitando l'evento *CurrentCellChanged* del *DataGrid* che mostra l'elenco delle mail, passiamo lo stesso *DataGrid* alla metodo *GetCurrentBindedObject* che si occuperà di estrarre dal *DataSet* associato i dati che ci interessano. Ma perché passare dal *GetCurrentBindedObject* invece di recuperare i dati direttamente dal *DataGrid?* Sebbene in questo caso avremmo potuto farlo, conviene sempre prelevare i dati dal *DataSet* associato in quanto il *DataGrid* potrebbe nascondere alcune delle colonne del *DataSet.* Ritornando al codice dell'evento *CurrentCellChanged,* possiamo vedere che, oltre a valorizzare il contenuto di alcune TextBox utili a visualizzare il contenuto delle mail, andremo a valorizzare anche un campo di tipo stringa denominato *mail_From:*

```
tbxFrom.Text = drv["From"].ToString();
mail_From = tbxFrom.Text;
```

che sarà esattamente quello che passeremo al metodo *SegnalaMail()* della classe proxy:

```
try{
  SpamManager sManager = new SpamManager();
  if ( sManager.SegnalaMail(Email) == 1 )
  {
    rtbLog.AppendText("L\'indirizzo "+Email+" è stato
                                   segnalato. Grazie\r\n");
  } else {
  rtbLog.AppendText("Si è verificato un problema durante
                                  la segnalazione\r\n");
}
```

LA PERSISTENZA DEI DATI

Il nostro client ora ha tutte le funzionalità che ci ser-

vono per scaricare i messaggi di posta, filtrarli e, eventualmente, farci segnalare nuovo Spam. Ma manca ancora qualcosa. Scegliendo il sistema di memorizzazione dei dati, abbiamo preferito un DataSet ad un Data Base. A differenza di un Data Base però, l'istanza di un DataSet "vive" finché è mantenuto un riferimento ad essa. Dopo di che, il Garbage Collector libererà la memoria ed i dati andranno persi. Come facciamo quindi a "salvare" il nostro DataSet per ritrovarlo integro ad ogni riavvio dell'applicazione? Sfrutteremo una funzionalità estremamente comoda fornita dal .Net Framework: la serializzazione. Per serializzazione si intende la possibilità di "trascrivere" su disco lo stato di un oggetto per poterlo poi ricreare attraverso il processo inverso (deserializzazione). Esistono diversi tipo di serializzazione in funzione del formatter scelto (BinaryFormatter, SoapFormatter, XmlSerializer). Per i nostri scopi utilizzeremo il BinaryFormatter che risulta essere semplice da usare e abbastanza veloce nelle operazioni di serializzazione e deserializzazione. Come prima cosa quindi creiamo una istanza del Formatter che vogliamo utilizzare:

```
System.Runtime.Serialization.Formatters.Binary.BinaryFor
        matter bin = new System.Runtime.Serialization
                .Formatters .Binary.BinaryFormatter();
```

Dovendo salvare i dati sul nostro Hard Disk, predisponiamo uno StreamWriter chiamato dat che scriva il dato nella cartella Mails contenuta nella stessa cartella di esecuzione del programma:

```
System.IO.StreamWriter dat = new StreamWriter(
    Environment.CurrentDirectory+@"\Mails\mails.dat");
```

Ora non ci resta che chiamare il metodo Serialize() del formatter a cui passeremo lo stream da serializzare e l'oggetto che vogliamo venga serializzato.

```
bin.Serialize(dat.BaseStream, dsmail);
dat.Close();
```

Nella directory Mails troveremo ora un nuovo file chiamato mails.dat che conterrà la rappresentazione binaria del nostro DataSet. Per recuperarlo dobbiamo avviare il processo inverso, chiamato appunto deserializzazione.

```
System.Runtime.Serialization.Formatters.Binary.BinaryFor
        matter bin = new System.Runtime.Serialization
                .Formatters.Binary.BinaryFormatter();
StreamReader reader_dsMail = new StreamReader(
    Environment.CurrentDirectory+@"\Mails\mails.dat");
DataSet d = (DataSet)bin.Deserialize(
                        reader_dsMail.BaseStream);
reader_dsMail.Close();
```

Lo stesso principio è stato utilizzato per la persistenza dei dati relativi alla casella di posta da controllare ed ai dati di login al server antispam solo che, al posto di serializzare un DataSet, è stato serializzato un normale oggetto.

UTILIZZO DEL CLIENT

Alla prima esecuzione del client di posta viene chiesto di impostare i dati di accesso sia al server POP3 che al server AntiSpam (vedi **Figura 4**). Per recuperare questo parametro fate riferimento al provider che fornisce la casella di posta elettronica. UserName e Password sono quelle che abitualmente usate per leggere la posta. Nella scheda "settaggi server antispam" bisogna inserire i dati di accesso al servizio (di default UserName = User e Password = User). Una volta inseriti questi dati sarà possibile utilizzare il client. Sulla barra di sinistra (vedi **Figura 5**) sono presenti una serie di icone, ognuna delle quali avvia un comando specifico. Ogni comando si attiva con un doppio click sull'icona relativa. Per scaricare la mail quindi, sarà sufficiente fare doppio click sull'icona denominata "Scarica Mail". Le mail scaricate con questo client NON verranno cancellate dal server, ne tantomeno questa funzionalità è prevista. Questa scelta è dovuta dal fatto che il client, sebbene funzionante, deve essere usato a scopo didattico. La cancellazione automatica delle mail scaricate potrebbe rendere poi impossibile il recupero di alcuni dati, pertanto non è stata implementata.

Fig. 4: *Impostazione dei dati di accesso al servizio*

Fig. 5: *Barra dei comandi del client di posta*

CONCLUSIONI

Si conclude qui la serie di 2 articoli sul sistema AntiSpam. Nel primo articolo abbiamo realizzato un server per la gestione ed il rilascio delle definizioni ed in questo abbiamo realizzato il nostro piccolo client capace di comunicare con il server e gestire le mail. Ho scelto di mantenere semplice a livello implementativo sia il server sia il client in modo da dedicare più spazio alla trattazione della logica che c'è dietro un piccolo sistema antispam. In questi 2 articoli abbiamo toccato argomenti come WSE 2, POP3, Sockets, Serializzazione ecc. vedendo quanto sono potenti questi strumenti quando lavorano insieme. Buon lavoro.

Michele Locuratolo

L'autore può essere contattato attraverso il suo blog su http://blogs.mindbox.it e sarà lieto di rispondere alle domande dei lettori.

Le classi Caching per accessi a velocità da record

Creiamo siti velocissimi

Vedremo come utilizzare gli oggetti messi a disposizione del namespace *Caching* per ottenere risposte più rapide da parte del nostro sito ASP.NET

Supponiamo che abbiate appena terminato di sviluppare una fantastica applicazione Web che fa uso di database. Ad esempio nella vostra applicazione Web potrebbe esserci una pagina che stampa a video l'elenco dei dipendenti di un'azienda, reperendo il contenuto da un database. Va da se che ogni volta che qualcuno accede alla pagina si scatenano almeno le seguenti operazioni

1) *Accesso al database*
2) *Esecuzione di una query*
3) *Stampa a video del risultato della query*

È un metodo classico e funziona bene da decenni. È anche vero che si spera che l'elenco dei dipendenti sia sufficientemente stabile, perciò la pagina restituita sarà più o meno sempre identica. Inoltre possono esserci più utenti che accedono contemporaneamente a questa pagina e ciascuno di loro genera un accesso al database e gli vengono ritornati sempre gli stessi dati. Perciò l'accesso a un database è un'operazione "eccessiva" per questo tipo di informazione. Nonostante questo, un database è sempre fondamentale per gestire applicazioni di tipo gestionale. Chiaramente abbiamo semplicemente fatto un esempio, ma possono esserci centinaia di casi in cui i dati non variano sufficientemente in fretta per giustificare un accesso continuo a un database, o comunque a una risorsa che li contiene. In tutti questi casi si può utilizzare una cache per ottimizzare le prestazioni.

L'IDEA

Banalmente l'idea è l'utilizzo di una "Cache". Una Cache è uno spazio comune condiviso da tutti gli utenti, in cui vengono copiati dei dati statici, frutto di una prima elaborazione. Nel caso dell'esempio precedente, in conseguenza al primo accesso alla pagina Web, il risultato statico dell'elaborazione potrebbe essere copiato in una cache. In questo modo tutte le interrogazioni future preleverebbero i dati statici dalla cache evitando la connessione al database ed ottimizzando così le prestazioni. Ovviamente deve esistere anche un qualche metodo tale che se il contenuto del database dovesse cambiare, la cache dovrebbe essere cancellata e ricostruita. In .NET esistono oggetti che consentono di applicare la tecnica descritta, e non solo vedremo che esistono metodi piuttosto avanzati, tali che il contenuto di una cache può essere modificato in base ad una serie di eventi e non solo la modifica di un dato. Ad esempio la cache potrebbe essere cancellata e ricostruita ad intervalli regolari di tempo, e molto altro ancora.

L'OGGETTO CACHE

L'oggetto *Cache* introdotto con ASP.NET rappresenta una delle più interessanti novità per gli sviluppatori. Simile all'oggetto Application, permette di memorizzare dati condivisi tra tutte le sessioni attive del proprio sito, ne migliora le caratteristiche aggiungendo tre modalità per il contenuto *Cache* viene rimosso dalla memoria:

1. **Modalità basata sul tempo:** specificando un certo numero di minuti o secondi è possibile rimuovere i dati dalla cache.
2. **Modalità basata su una chiave:** specificando una coppia *chiave/valore* è possibile rimuovere i dati dalla cache quando il contenuto di questo oggetto viene cambiato.

3. **Modalità basata su un file:** specificando il nome di un file con il relativo percorso, l'oggetto cache associato viene rimosso quando il file associato è modificato.

Queste modalità sono attivabili singolarmente attraverso il costruttore di un oggetto *CacheDependency*. Vediamo un esempio:

```
Dim dipendenza As New Caching.CacheDependency(
                                        "C:\file.dep")
```

In questo caso il costruttore utilizzato accetta solo un parametro, ovvero il nome del file dal quale l'oggetto *cache* dipende. L'oggetto inserito nella cache che utilizza questo oggetto di dipendenza sarà mantenuto in memoria fino a quando il file specificato non sarà modificato.

L'ESEMPIO COMMENTATO

Nel box laterale vediamo un esempio completo dove una pagina web utilizza l'oggetto *Cache* per memorizzare un *DataSet* riempito con il contenuto della tabella *Products* del database *Northwind* presente all'interno di Microsoft SQL Server. Inizialmente viene creato un oggetto per indicare la dipendenza della cache dal file chiamato *"file.dep"*. Successivamente viene controllata la *Cache* cercando la chiave *Prodotti* al suo interno. Se questa non esiste, ovvero è *Nothing*, vuol dire che questo è il primo accesso alla pagina oppure che il contenuto del file è stato modificato. In questo caso il data adapter riempie il *DataSet* con il contenuto della tabella *Products* e il tutto viene passato alla griglia. Infine il metodo *Insert* della *Cache* viene utilizzato per inserire il *DataSet* all'interno della cache. Come si può notare il metodo *Insert* accetta tre parametri: il nome dell'elemento che stiamo per aggiungere, l'oggetto da memorizzare e la dipendenza. Se la cache contiene già un elemento chiamato *"Prodotti"* allora il codice non fa altro che utilizzarla per recuperare il valore e fornirlo alla griglia. L'esempio non è certo indicato per dimostrare come la tecnica della cache sia utile per velocizzare la risposta del server. In questo caso la tabella contiene pochi record, il database è in locale e non scatena traffico di rete, un solo utente accede al database, ecc. Ma già con queste condizioni si ha un piccolo guadagno che testimonia come l'utilizzo della cache migliori le performance della propria applicazione. Cambiando il

1 **Creiamo con Visual Studio .NET un'applicazione web ASP.NET in VB.NET**

2 **Rinominiamo la pagina in Default.aspx**

3 **Trasciniamo dalla Toolbox un controllo di tipo SqlDataAdapter. Attraverso il wizard che compare specifichiamo la stringa di connessione verso il server che contiene il database Northwind. Infine, con il QueryBuilder del wizard, selezioniamo la tabella Products specificando di voler ricavare tutti i suoi record. L'istruzione SQL che dovrebbe risultare è *SELECT Products.** FROM Products. Rinominiamo l'oggetto SqlDataAdapter1 in daNorthwind.**

4 **Aggiungiamo un nuovo file di tipo DataSet e, tramite la finestra Server Explorer, dalla voce Data Connections selezioniamo il database Northwind a cui la nostra stringa di connessione fa riferimento e trasciniamo la tabella Products all'interno del DataSet appena creato. Questo farà in modo da creare lo schema del DataSet perfettamente uguale alla struttura della tabella Products del database Northwind.**

5 **Trasciniamo dalla Toolbox un controllo di tipo DataSet specificando di utilizzare il file DataSet creato al punto 4. Rinominiamo poi il DataSet in dsProducts.**

6 **Aggiungiamo una *DataGrid* chiamandola *dgNorthwind*; all'interno delle sue proprietà selezioniamo *Columns* e deselezioniamo la spunta per creare automaticamente le colonne. Noi vogliamo, infatti, aggiungere solo due colonne, il nome del prodotto e il suo prezzo. Percui aggiungiamo due *Bound Column:* la prima sarà collegata al campo della tabella chiamato *ProductName* e avrà l'intestazione *Nome,* la seconda sarà collegata al campo della tabella chiamato *UnitPrice* e avrà l'intestazione *Prezzo.***

7 **Aggiungiamo una label chiamata *lbTempo*; questa sarà utilizzata dal programma per**

visualizzare il tempo necessario a ricavare i dati dal database e caricare la griglia con e senza l'utilizzo della Cache.

8 **Creiamo un file di zero byte sotto la directory C: chiamandolo file.dep.**

```
Private Sub Page_Load(ByVal sender As
                 System.Object, ByVal e As
                        System.EventArgs)
Handles MyBase.Load
  Dim dipendenza As New Caching.
```

```
    CacheDependency("C:\file.dep")
  Dim inizio, fine As Long
  Dim tempo As TimeSpan
  inizio = DateTime.Now.Ticks
  If Cache("Prodotti") Is Nothing Then
    daNorthwind.Fill(dsProducts)
    dgNorthwind.DataSource =
                          dsProducts
    dgNorthwind.DataBind()
    Cache.Insert("Prodotti",
              dsProducts, dipendenza)
  Else
    dgNorthwind.DataSource =
                   Cache("Prodotti")
    dgNorthwind.DataBind()
  End If
  fine = DateTime.Now.Ticks
  lbTempo.Text = "Impiegato " & _
  tempo.FromTicks(fine-inizio).Milliseconds
          .ToString() & _ " millisecondi"
 End Sub
```

9 **All'interno del gestore dell'evento Load della pagina aggiungiamo il codice**

contenuto del file *"file.dep"* provocheremo la perdita del *DataSet* all'interno della cache. Utilizzando un'altra forma del metodo *Insert* che accetta sette parametri è possibile specificare il nome di una funzione *callback* che viene chiamata dall'applicazione quando l'oggetto è tolto dalla *Cache*. In questo modo potremmo inserire del codice che carica i nuovi dati all'interno del *DataSet* e aggiornare la cache inserendo nuovamente l'oggetto.

LA DIPENDENZA DI SQL SERVER

Un grosso limite della funzionalità di *Cache* di ASP.NET riguarda la mancanza totale di integrazione con un database. Sarebbe utilissimo, ad esempio, rimuovere un oggetto dalla *Cache* quando il contenuto di una tabella in un database viene modificato. Con l'uscita di Microsoft SQL Server 2005 e ASP.NET 2.0 questa lacuna sarà colmata ma se vogliamo aggiungerla adesso alla nostra applicazione come possiamo fare? Un modo semplice e abbastanza funzionale se utilizzato in un piccolo/medio contesto è quello di aggiungere un trigger all'interno del database e associarlo alla tabella di cui vogliamo monitorare i cambiamenti. Un *trigger* è una sorta di funzione *callback* che esegue delle istruzioni SQL quando i record di una tabella sono aggiunti, modificati e rimossi. Si può implementare un trigger in modo da modificare un file collegato ad un oggetto nella cache quando una tabella viene modificata nei suoi record. Vediamo come creare questo tipo di *trigger:*

1. Apriamo SQL Server Explorer espandendo rami della *treeview* fino a trovare il database Northwind e le sue tabelle.

2. Tramite la pressione del tasto destro del mouse facciamo comparire il menu dove scegliere la voce *All Tasks | Manage Triggers…*
   La dialog box comparirà permettendoci di creare un trigger associato alla tabella *Products.*

RICAVIAMO I DATI

Attraverso il wizard specifichiamo la stringa di connessione verso il server che contiene il database Northwind.
Infine, con il QueryBuilder del wizard, selezioniamo la tabella Products specificando di voler ricavare tutti i suoi record.
L'istruzione SQL che dovrebbe risultare è SELECT Products.* FROM Products. Rinominiamo l'oggetto SqlDataAdapter1 in daNorthwind.
Seguite il tutorial per meglio comprendere questo passo

1 **Il wizard parte con la classica schermata informativa. È sufficiente cliccare su Next per iniziare a gestire i parametri.**

2 **Come nome del server scegliamo localhost. Scorriamo poi la casella relativa al database fino a selezionare "NorthWind"**

3. Scriviamo il seguente codice SQL nel trigger e premiamo il tasto *Apply* per confermare la creazione del trigger.

```
CREATE TRIGGER AggiornaFile ON [dbo].[Products]
FOR INSERT, UPDATE, DELETE
AS
exec master..xp_cmdshell 'copy c:\file.dep c:\file.new'
exec master..xp_cmdshell 'del c:\file.dep'
exec master..xp_cmdshell 'copy c:\file.new c:\file.dep'
exec master..xp_cmdshell 'del c:\file.new'
```

La stored procedure *xp_ cmdshell* contenuta nel database master permette di eseguire dei comandi *shell* come la copia o la cancellazione di un file. In questo esempio viene chiamata per modificare il timestamp del file *"file .dep"*.

4. Eseguiamo la nostra applicazione web creata precedentemente.

5. Premendo il tasto di refresh del browser utilizziamo il DataSet inserito nella cache.

6. Da SQL Server Enterprise selezioniamo la tabella *Products* e con il tasto destro del mouse scegliamo la voce *Open Table | Return all rows…*

7. Cambiamo il *ProductName* del primo record da *Chai* a *Chai Ciao.*

8. Premiamo il tasto *refresh* del browser e vediamo come il nuovo record appare modificato all'interno della griglia.

Abbiamo creato un semplice sistema per aggiornare la cache quando cambia il contenuto di una tabella.

CONCLUSIONI

Il metodo esposto, seppur valido in un piccolo/medio contesto, ha delle limitazioni di cui bisogna tener conto.
Ad esempio, quando più utenti aggiornano la tabella associata al trigger ci potrebbero essere dei casi in cui non si riesca a modificare il file perché è in uso da un altro processo che lo sta già modificando. Per questo problema occorre implementare una serie di *lock/unlock* della tabella per permettere ad ogni modifica di generare un file modificato.

Fabio Claudio Ferracchiati

COS'È UNA FUNZIONE CALLBACK

**Nella vita reale, quando dovete portare a termine un compito, è più logico informare chi di dovere di aver terminato rispetto al fatto di essere continuamente tampinati con la domanda: hai finito?
Una funzione callback è una speciale funzione che viene chiamata direttamente dal sistema operativo quando ha finito o sta per iniziare un compito. Non occorre interrogare Windows ogni intervallo di tempo per sapere se, ad esempio, qualcuno ha premuto un tasto.
Sarà Windows a spedire un messaggio di tipo WM_KEYDOWN che, tramite una callback, può essere intercettato e gestito dal nostro programma.
Inoltre, gli eventi, che siamo abituati a trattare all'interno delle nostre applicazioni .NET, non sono altro che delle funzioni callback chiamate dal framework.**

3 **Recuperemo i dati utilizzando una normale query SQL per cui è necessario selezionare la prima opzione.**

4 **La tabella che utilizzeremo è la "Products" per cui possiamo selezionarla opportunamente dalla finestra di dialogo**

5 **Infine è sufficiente selezionare il carattere Jolly "*" per ottenere una select su tutti i campi della tabella products**

Come creare aste online in un sito Web

Commercio Online con le Ebay JAVA API

Un'introduzione alle API che Ebay, famoso sito di aste online, mette a disposizione degli sviluppatori che si vogliono cimentare nello sviluppo di innovative soluzioni per il commercio elettronico

Utilizza questo spazio per le tue annotazioni

Ebay è il sito di aste online più famoso al mondo. Attraverso questo sito è possibile inserire degli oggetti e creare delle vere e proprie aste. L'asta ha una durata variabile che va da 1 giorno a 10 giorni . L'asta inoltre può avere un prezzo di riserva oppure un prezzo "Compra Subito" (ovvero dove è possibile comprare immediatamente l'oggetto ad un prezzo fisso). Facendosi un giro su *www.ebay.it* (o meglio ancora su *www.ebay.com*) possiamo farci un'idea di quale gigantesca realtà ruota attorno a questo sito. Inoltre non esiste soltanto la versione italiana, ma tantissime versioni di Ebay (attualmente sono disponibili 25 versioni per altrettante differenti nazioni). Lo staff di Ebay ha messo a disposizione degli sviluppatori delle librerie per poter creare delle applicazioni che interagiscano con la "piattaforma" di Ebay. In questo articolo andremo a vedere cosa viene messo a nostra disposizione per quanto riguarda lo sviluppo in Java.

AMBIENTE DI SVILUPPO

Sul sito dedicato agli sviluppatori Ebay (*http://developer.ebay.com/DevProgram/index.asp*) troviamo notizie, forum, tool e documentazione riguardanti le possibilità che vengono offerte dalla piattaforma di Ebay. Già da tempo grandi siti come Ebay, Amazon, Google hanno sviluppato delle librerie da integrare in programmi esterni per far utilizzare le loro piattaforme. Ebay a partire dal 2000 permetteva di utilizzare delle chiamate attraverso XML. Ora è possibile anche utilizzare SOAP per usufruire delle risorse messe a disposizione. Una volta registrati all'interno del sito per sviluppatori è possibile scaricare l'SDK per Java, la quale comprende anche tutta la documentazione di cui abbiamo bisogno. La registrazione per un singolo individuo comprende: 5000 chiamate alle API a giorno, possibilità di sviluppare e autocertificare un'applicazione (per 100 $), accedere ai tool e alla documentazione disponibile nel sito per gli sviluppatori. L'SDK per Java ha bisogno attualmente di Apache Axis 1.1 e Java 2 Standard Edition versione 1.4.x per i sistemi Windows 2000 e Linux Red Hat 9. Per meglio capire come sia strutturata la piattaforma che Ebay mette a disposizione degli sviluppatori possiamo visionare il *WSDL (Web Service Definition Language)* attuale presente all'indirizzo *http://developer.ebay.com/webservices/latest/eBaySvc.wsdl*. Quella che segue è una piccola parte del WSDL (non è possibile riportarlo tutto visto che è grande 1 Mega)

```
....
....
<xs:complexType name="GetItemRequestType">
  <xs:annotation>
    <xs:documentation>
Contains the inputs that control what item data is
        retrieved. This includes the item ID that uniquely
          identifies the listing for which to retrieve data.
    </xs:documentation>
  </xs:annotation>
  <xs:complexContent>
    <xs:extension base="ebl:AbstractRequestType">
      <xs:sequence>
      <xs:element ref="ebl:ItemID">
      <xs:annotation>
      <xs:documentation>
  Specifies the item ID that uniquely identifies the
              item listing for which to retrieve the data.
                           ItemID is a required input.
          </xs:documentation>
          </xs:annotation>
          </xs:element>
        </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
....
```

Questa porzione di WSDL ci mostra come è stato

definito un tipo complesso che viene poi utilizzato facendo delle chiamate SOAP. In particolare vediamo dall'annotazione associata al tipo che un ID è un numero unico sulla piattaforma Ebay. Infatti l'ID rappresenta una determinata asta e avendo a disposizione questo identificativo (ad esempio in un nostro programma residente sul computer) possiamo avere tutte le informazioni relative a questa asta. Ritornando all'ambiente di sviluppo che ci viene fornito, possiamo installarlo su piattaforma Windows o Linux con due diversi installer. Una volta scaricato e installato avremo a disposizione tutta la documentazione necessaria per capire cosa ci troviamo davanti, le librerie da includere nel nostro progetto e molti esempi.

PRIMI PASSI: L'AUTENTICAZIONE

Come già detto esistono due modi per poter interagire con Ebay, due diverse tipologie di API: XML API e SOAP API. Noi in questo articolo andremo ad usare le SOAP API. Esistono diverse operazioni che possiamo effettuare. In generale la classica transazione è composta di 4 semplici passi: preparazione della richiesta, invio, ricezione della risposta e decodifica. Praticamente tutte le funzioni che richiamiamo all'interno di Ebay sono strutturate in questa maniera. Prima di iniziare a vedere qualsiasi esempio bisogna vedere come effettuare l'autenticazione con il sistema di Ebay. L'autenticazione infatti è alla base di ogni importante feature che ci viene concessa con le Ebay API. Come già detto in precedenza dobbiamo avere un utente registrato per poter iniziare a testare le nostre applicazioni. In questa fase dello sviluppo serve avere un utente iscritto nel SandBox Enviroment. Per poter iscrivere un nuovo utente dobbiamo prima di tutto collegarci all'indirizzo *http://developer.ebay.com/DevZone/getstarted/setup.asp*. Il primo passo da compiere è riempire i dati relativi al nuovo utente. Una volta registrato bisogna fare la verifica dell'utente, scrivendo il primo programma che utilizza le API Ebay. Praticamente ci sarà bisogno di scrivere un programma che richiami la funzione *ValidateTestUserRegistration*, che invia tutti i dati che vi sono stati dati in fase di registrazione e vi permette di essere un utente abilitato ad interagire con il SandBox Enviroment. Ora che abbiamo un utente registrato scriviamo la parte di Autenticazione che inseriremo nei nostri programmi (magari creando un opportuna classe che svolge questo particolare compito). Quando vogliamo autenticarci con il sistema di Ebay riceviamo un Token, il quale ci permette di effettuare più di un chiamata verso le Ebay API. Ecco quindi un semplice esempio di inizializzazione dei parametri riguardanti l'autenticazione

```
ApiContext apiContext = new ApiContext();
ApiCredential cred = apiContext.getApiCredential();
ApiAccount ac = cred.getApiAccount();
eBayAccount ec = cred.geteBayAccount();
input = ConsoleUtil.readString("Inserisci API Developer ID: ");
ac.setDeveloper(input);
input = ConsoleUtil.readString("Inserisci API
                                        Application ID: ");
ac.setApplication(input);
input = ConsoleUtil.readString("Inserisci API Certificate: ");
ac.setCertificate(input);
input = ConsoleUtil.readString("Inserisci il tuo ID Ebay: ");
ec.setUsername(input);
input = ConsoleUtil.readString("Inserisci la password
                                        del tuo account: ");
ec.setPassword(input);
```

Come possiamo vedere i parametri che dobbiamo inserire sono molti, praticamente tutti quelli in nostro possesso dopo la registrazione. Attraverso i metodi statici della classe *ConsoleUtil* possiamo tranquillamente provare l'applicazione, senza costruire un'interfaccia grafica. È importante da ricordare che il Token che ci viene assegnato nell'autenticazione è provvisorio, quindi se la nostra applicazione prevede diversi collegamenti al sistema di Ebay dobbiamo implementate l'intefaccia *TokenEventListener*. Questa notificherà l'evento riguardante l'invalidazione del Token e quindi sarà nostro compito effettuare nuovamente l'autenticazione.

Fig. 1: *Architettura della piattaforma Ebay*

AGGIUNGERE UN OGGETTO

Ora che ci siamo identificati con il sistema di Ebay im-

NOTA

Esiste un programma disponibile sul sito di eBay, Turbo Lister, che permette di inserire le aste e monitorarle. Questo tool è utile soprattutto per venditori che hanno molte aste e hanno il bisogno di inserire tanti oggetti in maniera veloce.
http://pages.ebay.com/turbo_lister

AMBIENTE DI TEST

Ebay non poteva permettersi di dare come campo di battaglia per testare le applicazioni il vero e proprio sistema per aste online. In questo modo si sarebbero venuti a creare dei problemi assurdi riguardanti la sicurezza e la serietà dei contenuti dei vari siti. Proprio per questo motivo sono state definite due diverse modalità: *Sandobox* e *Production*. Quest'ultima rappresenta il reale sistema di aste online, dove in ogni momento c'è un'asta che scade, un nuovo utente che si registra etc. etc. *Sandbox* invece è un *Environment* creato per testare le applicazioni degli sviluppatori. In questa "scatola di sabbia" (come quella relativa alle Applet e MIDlet) noi possiamo creare nuovi utenti fittizi, avere a disposizione un finto budget per comprare oggetti inesistenti. All'indirizzo http://sandbox.ebay.com troviamo tutte le informazioni necessarie per poter iniziare a interagire con questo sistema. Una volta che poi la nostra applicazione sarà stata testata, per utilizzarla sulla reale piattaforma di Ebay non dovremo far altro che cambiare i vari indirizzi ai quali facciamo riferimento nel nostro programma.

pariamo come inserire un oggetto in vendita attraverso le API Java. Questa funzione può essere molto utile a negozi o privati che vogliono vendere molti oggetti su Ebay e vogliono quindi evitare di riempire ogni volta tantissime form ma automatizzare tutto il processo. Come prima cosa dobbiamo dire al nostro programma qual è l'URL da richiamare per le richieste SOAP

```
input = ConsoleUtil.readString("Enter eBay SOAP
                                        server URL: ");
apiContext.setApiServerUrl(input);
```

L'inserimento di un oggetto è un compito abbastanza semplice che vediamo riportato nelle seguenti righe di codice

```
ItemType item = new ItemType();
//Settiamo la tipologia di asta, in questo caso Chinese,
//che significa un'asta singola per altre tipologie di aste
//vi rimando alla documentazione ufficiale
item.setListingType(ListingTypeCodeType.Chinese);
//Qui settiamo la valuta con la quale viene listata la
                                              nostra asta
item.setCurrency(CurrencyCodeType.USD);
item.setCountry(CountryCodeType.US);

//Esistono diversi metodi di pagamento che un
//venditore può accettare per un asta. In questo caso
//inseriamo la generica descrizione che dice di
//guardare la descrizione dell'oggetto
item.setPaymentMethods(new
   BuyerPaymentMethodCodeType[] {BuyerPaymentMethod
                      CodeType.PaymentSeeDescription} );
item.setRegionID("0");
item.setListingDuration(ListingDurationCodeType.Days_3);
//Ora bisogna settare le informazioni importanti
//riguardanti il titolo, la descrizione, il luogo, la
//quantità e il prezzo di partenza
item.setTitle(ConsoleUtil.readString("Titolo: "));
item.setDescription(ConsoleUtil.readString("Descrizione: "));
item.setLocation(ConsoleUtil.readString("Luogo: "));
input = ConsoleUtil.readString("Quantità: ");
item.setQuantity(new Integer(input));
input = ConsoleUtil.readString("Prezzo di partenza: ");
item.setStartPrice(new AmountType(
                    new Double(input).doubleValue()));

//Settaggio della categoria dove inseriamo la nostra asta.
//Per la lista completa degli ID delle categorie
//possiamo consultare la documentazione
CategoryType cat = new CategoryType();
cat.setCategoryID(ConsoleUtil.readString(
            "Categoria primaria (ad esempio 357): "));
item.setPrimaryCategory(cat);
```

In questo modo abbiamo preparato il nostro oggetto per poterlo inserire in un asta Ebay (chiaramente quando facciamo le prove nella SandBox di Ebay). Per poterlo vedere tramite browser dobbiamo chiaramente sapere qual'è l'ID con il quale è stato inserito dal sistema. Questo lo possiamo sapere quando effettivamente inseriamo l'asta

```
AddItemCall api = new AddItemCall(apiContext);
FeesType fees = api.addItem(item);
double listingFee = eBayUtil.findFeeByName(
      fees.getFee(), "ListingFee").getFee().getValue();
System.out.println("Il prezzo dell'inserzione è :
              " + new Double(listingFee).toString());
System.out.println("L'ID dell'inserzione è :
                  " + item.getItemID().getValue());
```

Collegandoci al sito *http://cgi.sandbox.ebay.com/ws/eBayISAPI.dll?ViewItem&item=NOSTROID* possiamo ora vedere l'asta online.

NOTA

Java non è l'unico linguaggio che ci permette di interagire con il sistema di eBay. In qualsiasi linguaggio possiamo richiamare le API XML, chiaramente interfacciandosi con le opportune librerie per manipolare XML. Esiste inoltre una versione dell'SDK per lo sviluppo in linguaggi .NET
http://developer.ebay.com/windows/

Amazon, altro famoso sito di eCommerce, permette agli sviluppatori e ai webmaster di usufruire dei contenuti del proprio sito. Gli AWS (Amazon Web Services) danno la possibilità di interagire con il sistema di Amazon in maniera abbastanza simile a quella che viene offerta da eBay.
http://www.amazon.com

DOWNLOAD ED INSTALLAZIONE DELL'EBAY JAVA SDK

1 **Dobbiamo prima di tutto scaricare la versione dell'SDK per Java dal sito http://developer.ebay.com, per poi procedere con l'installazione, con il classico wizard.**

CREAZIONE DELLO USER

2 **Prima di poter creare un qualsiasi programma, che faccia delle richieste verso il sistema di Ebay, dobbiamo registrare uno user che possa fare i test nella Sandbox Environment di Ebay. Quindi dobbiamo attivare quest'utente sul sito http://developer.ebay.com.**

CLASSPATH DELLE LIBRERIE

3 **Per poter utilizzare le librerie che sono incluse nell'SDK dobbiamo includerle nel nostro classpath. Le possiamo trovare nella sottocartella lib della directory dove abbiamo installato l'SDK. Come sempre possiamo definirle o a livello di sistema oppure nel nostro ambiente di lavoro preferito.**

RICERCA OTTIMIZZATA

Sarebbe interessante implementare un motore di ricerca tipo Froogle (*www.froogle.com*).
Grazie a questo noi possiamo avere informazioni riguardanti i prezzi di un oggetto che cerchiamo su diverse siti di ecommerce.
La stessa cosa noi la possiamo implementare sul sistema di Ebay. Quello che dobbiamo fare è effettuare diverse ricerche su diversi siti Ebay. Come sempre istanziamo la classe che vogliamo utilizzare passando come parametro l'*ApiContext* dell'applicazione

```
GetSearchResultsCall api = new GetSearchResultsCall(
                                           apiContext);
GetSearchResultsRequestObjectType req[] =
          new GetSearchResultsRequestObjectType[3];
```

Abbiamo inizializzato tre diversi oggetti per la ricerca perché effettueremo tre diverse ricerche e le riporteremo in un'unica schermata. Ora dobbiamo iniziare a settare le informazioni che vogliamo richiedere ad Ebay.

```
PaginationType p = new PaginationType();
p.setEntriesPerPage(50);
p.setPageNumber(1);
for (int i=0;i<3;i++) {
req[i].setQuery("Geek phone");
req[i].setPagination(p);
}
```

In questo modo abbiamo dichiarato la query che effettueremo sul sistema di Ebay e il numero di entry per pagina (50). Ora che siamo usciti dal ciclo for dobbiamo specificare per ogni diversa richiesta il sito su cui andare a cercare, raffinando quindi la nostra ricerca.

```
SearchLocationFilterType slft[]=
                           new SearchLocationFilterType[3];
SearchLocationType sltUK=new SearchLocationType();
SearchLocationType sltUS=new SearchLocationType();
SearchLocationType sltIT=new SearchLocationType();
sltUK.setSite(SiteCodeType.UK); //Inghilterra
sltUS.setSite(SiteCodeType.US); //Stati Uniti
sltIT.setSite(SiteCodeType.Italy); //Italia
slft[0].setSearchLocation(sltUK);
slft[1].setSearchLocation(sltUS);
slft[2].setSearchLocation(sltIT);
req[0].setSearchLocationFilter(slft[0]);
req[1].setSearchLocationFilter(slft[1]);
req[2].setSearchLocationFilter(slft[2]);
```

E alla fine dobbiamo semplicemente eseguire tutte e tre le richieste che abbiamo definito.

```
SearchResultItemType[] theItemListingsUK =
                           new SearchResultItemType[];
SearchResultItemType[] theItemListingsUS =
                           new SearchResultItemType[];
SearchResultItemType[] theItemListingsIT =
                           new SearchResultItemType[];
theItemListingsUK = api.getSearchResultsCall(req[0]);
theItemListingsUS = api.getSearchResultsCall(req[1]);
theItemListingsIT = api.getSearchResultsCall(req[2]);
```

Dai risultati che otteniamo, *SearchResultItemType*, dobbiamo estrarre i vari oggetti e successivamente possiamo rappresentarli a schermo in diverse modalità. Se inseriremo questo programma in una servlet dovremo formattare una pagina web che contenga questi risultati.
Altrimenti potremmo utilizzarli su un programma standard J2SE, costruendo un'opportuna interfaccia grafica.

Federico Paparoni

eBay mette a disposizione dello sviluppatore anche dei forum e una newsletter, con la quale vengono periodicamente notificate ai programmatori le ultime novità riguardanti l'evoluzione del sistema. Inoltre viene data la possibilità di inserire i programmi sviluppati in un negozio (eBay Solutions Directory), dove possiamo trovare programmi e soluzioni sviluppate da eBay stesso o da sviluppatori come noi. Questa è chiaramente un'ottima vetrina per gli sviluppatori che vogliono proporre soluzioni innovative riguardanti l'eCommerce.
http://developer.ebay.com/DevZone/community/index.asp
http://developer.ebay.com/DevZone/account/SDMarketing.asp

AUTENTICAZIONE

```
ApiContext apiContext = new
ApiContext();
ApiCredential cred =
apiContext.getApiCredential();
ApiAccount ac = cred.getApiAccount();
eBayAccount ec = cred.geteBayAccount();
ac.setDeveloper(); //Developer ID
ac.setApplication(); //Application ID
ac.setCertificate(); //Certificato delle API
ec.setUsername(); //Username
ec.setPassword(); //Password
```

4 **L'autenticazione è d'obbligo per la fruizione di molte funzionalità nel sistema di Ebay. Per poterla effettuare dobbiamo avere a disposizione tutti i dati che ci sono stati restituiti dal sistema di Ebay nella fase di registrazione.**

CREAZIONE DI UN NUOVA ASTA

```
ItemType item = new ItemType();
item.setListingType(ListingTypeCodeType.
                                   Chinese);
item.setCurrency(CurrencyCodeType.USD);
item.setCountry(CountryCodeType.US);
...
item.setTitle(ConsoleUtil.readString("Titolo: "));
item.setDescription(ConsoleUtil.readString
                          ("Descrizione: "));
item.setLocation(ConsoleUtil.readString(
                                "Luogo: "));
input = ConsoleUtil.readString("Quantità: ");
item.setQuantity(new Integer(input));
```

5 **Per poter inserire un nuovo oggetto per creare un'asta online dobbiamo istanziare un nuovo oggetto. Utilizziamo la classe ItemType e settiamo i valori necessari.**

INSERIMENTO NUOVA ASTA

```
AddItemCall api = new AddItemCall(
                                  apiContext);
FeesType fees = api.addItem(item);
double listingFee = eBayUtil.findFeeByName(
        fees.getFee(),"ListingFee").getFee()
                                  .getValue();
System.out.println("Il prezzo
        dell'inserzione è : " + new Double(
                 listingFee).toString());
System.out.println("L'ID dell'inserzione è :
          " + item.getItemID().getValue());
```

6 **Per l'inserimento dell'oggetto dobbiamo ottenere l'ApiContext, dove abbiamo inserito tutte informazioni per l'autenticazione. L'oggetto viene passato come argomento a AddItemCall.**

In questa puntata: gestione delle collisioni tra mesh

Scontro tra mostri

IV parte

La corretta organizzazione delle collisioni tra mesh 3D è importante. IrrLicht è uno strumento potente, facile da usare e intuitivo che ci consente di implementarla al meglio

Utilizza questo spazio per le tue annotazioni

Uno degli aspetti fondamentali della programmazione di un videogioco 3D è il rilevamento delle collisioni tra le mesh. Per "rilevamento delle collisioni" si intende il riconoscimento di una situazione in cui due mesh sono talmente vicine da compenetrarsi. Come abbiamo già avuto modo di vedere, ogni motore 3D permette di posizionare una mesh in qualsiasi punto dell'ambiente che viene rappresentato. Questo vuol dire che, posizionando due mesh nello stesso identico punto, o comunque molto vicine, le loro figure appariranno "una dentro l'altra", dando un effetto di inconsistenza materiale. Insomma avremmo a che fare con due ologrammi! È compito del motore 3D, in questo caso di IrrLicht, accorgersi del fatto che due o più mesh si stanno compenetrando. È compito invece del programmatore utilizzare questa informazione per realizzare il comportamento voluto, ad esempio spostare le mesh che si scontrano in modo che non collidano più.

TIPI DI COLLISIONE

Rilevare le collisioni è un compito tutt'altro che banale. Mentre, infatti, chiunque di noi potrebbe dire se due mesh si stanno compenetrando semplicemente osservandole, farlo fare a un programma è molto più difficile. Un algoritmo deve ragionare in termini di *"vertici"* e *"triangoli"* e basarsi su considerazioni geometriche, anziché su percezioni visive come facciamo noi. Esistono tuttavia, per quanto complessi, degli algoritmi che permettono di effettuare questo calcolo con precisione matematica. Il problema più grande, quindi, non è trovare un procedimento che faccia questa cosa, quanto trovarne uno che la faccia bene! La vera difficoltà, infatti, risiede nel mantenere le performance di calcolo "in tempo reale". Gli algoritmi di *"collision detection"* sono molto pesanti e più sono complesse le mesh interessate, maggiore sarà il tempo di calcolo richiesto. Per superare questi problemi sono state sviluppate negli anni procedure sempre più ottimizzate, dalle prestazioni sempre migliori. IrrLicht implementa diversi di questi algoritmi nascondendone i particolari agli occhi dell'utente. Permette tuttavia di scegliere tra alcuni tipi di rilevamento delle collisioni a seconda della loro complessità e precisione. Sta infatti al programmatore stabilire se l'applicazione che sta sviluppando necessita di grande accuratezza oppure si può accontentare di una certa approssimazione in favore di prestazioni più elevate. I metodi di rilevamento delle collisioni sono sostanzialmente di due tipi:

- **metodi esatti:** viene considerata interamente la geometria delle mesh interessate alla collisione. Questi metodi sono precisissimi e funzionano bene con mesh di qualsiasi forma: convesse, concave, allungate ecc. Sono però molto lenti rispetto ad altri metodi e il loro tempo di calcolo aumenta all'aumentare della complessità della mesh.
- **metodi approssimati:** viene considerata una mesh più "semplice" di quella visualizzata. Ad esempio un parallelepipedo *(bounding box)* o una sfera (metodo della distanza) che "racchiudano" la mesh originale. Questi metodi non sono precisi in quanto un oggetto che collida con la mesh semplificata, di solito più grande, potrebbe non collidere con quella reale. Sono però metodi velocissimi per cui molto spesso conviene utilizzarli.

La giusta via da percorrere è quella di cercare un compromesso tra velocità e precisione. IrrLicht of-

fre gli strumenti per perseguire questo fine, vediamo come.

L'AMBIENTE DI PROVA

Vediamo di seguito come utilizzare un metodo approssimato per il calcolo della collisione tra videocamera e ambiente circostante. Come prima cosa scriviamo il codice per caricare un ambiente di test credibile. Ci affidiamo al nostro bel livello quake3-style, disponibile nella cartella *"media"* dell'istallazione di IrrLicht:

```
int main() {
  //Setup iniziali
  IrrlichtDevice *device = createDevice
  (EDT_DIRECTX8, dimension2d<s32>(640, 480), 16, false);
  if (device == 0)
    return 1; // impossibile creare il driver
  IVideoDriver* driver = device->getVideoDriver();
  ISceneManager* smgr = device->getSceneManager();
  //Carico il livello
  device->getFileSystem()
      ->addZipFileArchive("media/map-20kdm2.pk3");
  IAnimatedMesh* q3levelmesh = smgr->
                              getMesh("20kdm2.bsp");
  ISceneNode* q3node = 0;
  if (q3levelmesh)
    q3node = smgr->addOctTreeSceneNode(
                        q3levelmesh->getMesh(0));
  //Creo la videocamera
  ICameraSceneNode* camera = smgr->
            addCameraSceneNodeFPS(0,100.0f,300.0f);
  //Disabilito il cursore del mouse
  device->getCursorControl()->setVisible(false);
  //Ciclo principale
  while(device->run()) {
    driver->beginScene(true, true, 0);
    smgr->drawAll();
    driver->endScene(); }
  //Rilascio le risorse e esco
  device->drop();
  return 0; }
```

Fin qui nulla di nuovo: la struttura è quella classica di un programma con IrrLicht. Abbiamo creato il *device*, il *driver* e lo *SceneManager*. Abbiamo attaccato al filesystem la mesh del livello che vogliamo caricare e abbiamo creato un nodo con essa. Ci siamo sbarazzati dell'odioso puntatore del mouse e, per completare l'opera, abbiamo creato una videocamera di tipo FPS per poterci spostare nel livello. All'interno del ciclo principale ci occupiamo unicamente di disegnare l'intera scena. Provando a muoverci coi tasti direzionali otteniamo un effetto di "volo d'angelo": possiamo andare in qualsiasi direzione. Non siamo soggetti a gravità e oltrepassiamo allegramente i muri che, invece, dovrebbero essere solidi. Vediamo come ottenere un effetto decisamente più realistico sfruttando il sistema di rilevamento delle collisioni.

VIDEOCAMERA E METODO DELLA DISTANZA

Per rilevare le collisioni utilizzando le funzionalità offerte da IrrLicht è necessario creare un oggetto di tipo *TriangleSelector* da associare alla mesh che ci interessa:

```
ITriangleSelector* selector = 0;
if (q3node) {
  selector = smgr->createOctTreeTriangleSelector
  (q3levelmesh->getMesh(0), q3node, 128);
  q3node->setTriangleSelector(selector);
  selector->drop(); }
```

L'*ITriangleSelector* è la base per la *collision-detection* di IrrLicht: oggetti di questo tipo vengono utilizzati sia coi metodi esatti sia con quelli approssimati. Come già detto utilizzeremo un metodo di questo secondo tipo per la gestione della videocamera FPS. Quello che ci interessa è dare la sensazione di "ingombro" della videocamera, piuttosto che associarla a una mesh ben definita. Per questo motivo possiamo tranquillamente approssimarla con una bounding box o una sfera. In realtà IrrLicht implementa un algoritmo di approssimazione un po' più preciso, che permette di specificare un ellissoide anziché una sfera. Un ellissoide è una "ellisse in 3 dimensioni", una specie di "uovo" per il quale è necessario specificare 3 raggi, uno per ogni asse. Ovviamente se proprio dovessimo avere bisogno di una sfera, sarà sufficiente fornire 3 raggi uguali. Possiamo associare un particolare comportamento alla videocamera aggiungendo al suo nodo un opportuno animatore: un oggetto di tipo *ISceneNodeAnimator.* Questo oggetto si occupa di modificare il movimento della videocamera quando questa viene a collidere con l'ambiente circostante. In particolare viene modificata la traiettoria in maniera tale da non farle oltrepassare le mesh con cui collide. È questo comportamento che darà la sensazione di "solidità" dei muri del livello. L'animatore utilizzato in questo caso è di tipo *Col-*

Fig. 1: *Senza collisioni e gravità possiamo spostarci in qualsiasi punto dell'ambiente*

lisionResponseAnimator e consente di inserire anche un valore per l'accelerazione gravitazionale. In definitiva il codice completo per la gestione delle collisioni videocamera-mondo è il seguente:

```
ISceneNodeAnimator* anim = smgr->
      createCollisionResponseAnimator(selector, camera,
            vector3df(30,50,30), vector3df(0,-100,0),
                    100.0f, vector3df(0,50,0));
camera->addAnimator(anim);
anim->drop();
```

Fig. 2: **Tramite "picking" possiamo evidenziare esattamente il triangolo osservato**

Viene istanziato l'animatore, successivamente lo si aggiunge al nodo della videocamera (camera) e quindi lo si rilascia (*drop()*) in quanto sarà gestito automaticamente dallo *scene-manager.* Da notare come la stessa sorte sia toccata anche al *triangle-selector* istanziato in precedenza. I parametri della *createCollisionResponse-Animator()* specificano:

- l' *ITriangleSelector* della mesh con cui verificare la collisione;
- il nodo racchiuso dall'ellissoide (camera);
- i 3 raggi dell'ellissoide passati tramite un *"vector3df"*;
- il vettore dell'accelerazione di gravità;
- l'incremento della gravità per secondo;
- lo spostamento dell'ellissoide rispetto al centro della mesh.

Il più delle volte è sufficiente chiamare la *createCollisionResponseAnimator()* passando solo i primi due parametri, gli altri hanno valori ragionevoli di default. A questo punto potremo controllare la videocamera coi tasti direzionali e il mouse come abbiamo fatto sinora, ma il suo comportamento sarà modificato dall'animatore. Non potremo infatti più oltrepassare i muri o "volare" in quanto saremo mantenuti a terra dalla gravità. La cosa bella del meccanismo degli animatori è che è possibile applicarli a qualsiasi nodo della scena, non solo alla videocamera. Se vogliamo inserire una mesh soggetta a forza di gravità non dobbiamo fare altro che utilizzare la funzione *addAnimator()* analogamente a quanto fatto poc'anzi.

PICKING

Utilizzare il *CollisionResponseAnimator* è molto utile in fase di sperimentazione/debug oppure per applicazioni abbastanza semplici. In generale, tuttavia, il comportamento di un dato elemento andrà programmato in maniera più "fine", per conformarsi alle necessità dell'applicazione che si sta creando. Proprio per questo motivo in IrrLicht è disponibile un oggetto di classe *ISceneCollisionManager* che si occupa di gestire "a basso livello" le collisioni tra mesh. Questo collision-manager, ottenibile dallo *scene-manager* tramite la funzione *getSceneCollisionManager()*, fornisce informazioni interessanti. È possibile sapere, ad esempio, il punto esatto di contatto tra due mesh o il triangolo interessato dalla collisione. Manipolando opportunamente queste informazioni si possono ottenere comportamenti molto raffinati come, ad esempio, la creazione di un vero e proprio "motore fisico" che gestisca gli oggetti secondo la loro velocità, accelerazione, massa ecc. In questa sede faremo un esempio decisamente più didattico: faremo disegnare in rosso i bordi del triangolo "puntato"

BOUNDING BOX COMPLESSE

Spesso qualunque sia la scelta tra metodo esatto o approssimato per le collisioni il risultato sarà inaccettabile. Per questo sono stati sviluppati diversi algoritmi "intermedi" tra questi due estremi. È ad esempio possibile utilizzare una mesh semplificata preelaborata con un editor 3D e caricata da file. Oppure si possono associare più di una bounding box per ogni mesh complessa. Una figura 3D di un uomo potrà avere bounding box diverse per la testa, le braccia, il torso ecc. In questo modo si aumenta la precisione deteriorando in maniera trascurabile le prestazioni.

CHE COS'È?

1 **Il billboarding è la tecnica di inserire immagini 2D in una scena 3D per simulare la presenza di mesh complesse. Molto utilizzata in passato (i mostri di Doom erano 2D!) è usata oggi in maniera molto più raffinata e "discreta".**

COME SI USA?

```
// Creo un oggetto per il billboarding
IBillboardSceneNode * bill = smgr->
                    addBillboardSceneNode();
// Modifico parametri come trasparenza,
// immagine mostrata e dimensione
bill->setMaterialType(
               EMT_TRANSPARENT_ADD_COLOR );
bill->setMaterialTexture(0, driver->
             getTexture("ioprogrammo.bmp"));
bill->setMaterialFlag(EMF_LIGHTING, false);
bill->setSize(dimension2d<f32>(40.0f, 60.0f));
```

2 **IrrLicht supporta nativamente il billboarding attraverso la classe *IBillboardSceneNode.* Questa permette di visualizzare una immagine 2D e modificarne numerosi parametri.**

dalla videocamera. Per farlo utilizzeremo una tecnica elementare: creeremo una linea che congiunga la videocamera con il punto preciso che sta inquadrando. Di seguito chiederemo a IrrLicht di fornire il punto di intersezione di questa retta con la mesh del livello nonché il triangolo della mesh che lo contiene. Questo triangolo verrà infine disegnato a schermo, evidenziato in rosso. Il codice che realizza questo comportamento è il seguente:

```
//Preparo il materiale per il triangolo
SMaterial material;
material.Lighting = false;
...
// Codice per evidenziare il triangolo guardato
line3d<f32> line;
line.start = camera->getPosition();
line.end = line.start +
(camera->getTarget() - line.start).normalize() * 1000.0f;
vector3df intersection;
triangle3df tri;
if (smgr->getSceneCollisionManager()->
      getCollisionPoint(line, selector, intersection, tri)) {
    driver->setTransform(ETS_WORLD, matrix4());
    driver->setMaterial(material);
    driver->draw3DTriangle(tri, SColor(0,255,0,0));}
```

Come prima cosa viene istanziato un materiale fittizio *(SMaterial material)* cui viene disabilitata l'illuminazione. Questo serve in seguito quando viene chiamata la *draw3DTriangle()*. All'interno del ciclo principale del programma *(while(device->run()) {...})* viene creato un oggetto di tipo *line3d* che rappresenta un segmento nello spazio.
Gli estremi del segmento vengono così fissati:

- **inizio (line.start):** posizione della videocamera;
- **fine (line.end):** il punto "visto" dalla videocamera, che si trova a distanza 1000 da essa.

Viene poi chiamata la funzione *getCollisionPoint()* che prende in ingresso la linea creata *(line)* e il *ITriangleSelector* della mesh del livello *(selector)* e restituisce, se esistono, il punto di intersezione *(intersection)* e il triangolo che lo contiene *(tri)*. In caso di collisione rilevata le coordinate del triangolo vengono riportate in coordinate coerenti col mondo (matrice *ETS_WORLD*), viene disabilitata la luce impostando il materiale creato all'inizio e infine viene disegnato in rosso il triangolo trovato tramite la *draw3DTriangle()*. La tecnica di selezionare un oggetto in base a ciò che viene inquadrato è detta *"Picking"* ed è fondamentale in qualsiasi gioco 3D in circolazione. Si pensi solo a quando selezioniamo una unità di combattimento in giochi come *Command&Conquer* o a quando miriamo per sparare in un FPS qualsiasi. Il fatto che IrrLicht renda semplice una operazione come questa dimostra, se ancora ce ne fosse bisogno, quanto sia valido come motore, e non solamente a livello grafico.

CONCLUSIONI

Chiudiamo, con questo articolo, la serie dedicata a IrrLicht. Nel corso di queste 4 puntate abbiamo dato una rapida panoramica delle enormi potenzialità fornite da questo engine. Argomenti come la gestione della scena, il caricamento delle mesh o, da ultimo, la gestione delle collisioni, sono le basi per la creazione di un software interattivo tridimensionale e IrrLicht ne garantisce un utilizzo immediato ed efficace. La nostra avventura nel mondo della programmazione 3D non finisce però qui, per cui... restate sintonizzati su queste frequenze!

Alfredo Marroccelli

IL PUNTO ESATTO

```
// Ottengo il punto di intersezione col livello
vector3df intersection;
triangle3df tri;
smgr->getSceneCollisionManager()->
getCollisionPoint(line, selector, intersection, tri)

// Imposto il billboarding nel punto trovato
bill->setPosition(intersection);
```

3 Per visualizzare un'immagine in billboarding è necessario specificare un punto dell'ambiente 3D tramite la funzione setPosition(). Nel nostro caso prendiamo il punto "intersection", utilizzato nell'articolo.

IL RISULTATO FINALE

4 Il punto del billboarding è modificabile semplicemente muovendo il mouse. L'effetto è di avere una copia di ioProgrammo sempre davanti agli occhi, ma per i nostri lettori questa sarà una sensazione tutt'altro che nuova...

BILLBOARDING AVANZATO

L'effetto sgradevole del billboarding è dato dal fatto di dovere osservare sempre la stessa immagine per un oggetto, anche muovendoci attorno ad esso. Un modo per ovviare a questa situazione è quello di prevedere diverse immagini per la nostra mesh in billboarding, una per ogni angolazione plausibile. Nel videogioco di calcio FIFA'96, ad esempio, i giocatori erano ripresi da 8 diverse angolazioni. Venivano mostrati, col billboarding, scegliendo la gusta immagine in base alla loro direzione e alla posizione della videocamera che li inquadrava.

Persistenza dei dati con Java

Congela i tuoi oggetti con Hibernate!

Un potente tool open-source in grado di rendere persistenti, in modo del tutto trasparente, oggetti Java. Hibernate è semplice ed intuitivo e può essere utilizzato con i database relazionali più diffusi

Hibernate è un prodotto open-source, scritto in Java, che può essere utilizzato sia in contesti open-source sia in applicazioni commerciali. In pratica, esso funziona come una libreria separata da inserire nei propri progetti. L'API della libreria è semplice, intuitiva e ben documentata. Hibernate può essere scaricato dal sito *http://www.hibernate.org*. Il compito principale di Hibernate è quello di mappare oggetti Java in tabelle di un database relazionale. Nel più semplice dei casi, gli elementi in gioco sono quindi tre: l'*oggetto,* la *tabella* e un qualcosa che dica come trasferire il contenuto informativo dall'oggetto alla tabella. Chiameremo questo qualcosa *descrittore*. Ogni qualvolta che Hibernate accederà al database, sia per memorizzare oggetti sia per effettuare query, consulterà il descrittore che lo guiderà nel compiere l'operazione. Supponiamo di avere una classe *Articolo* di attributi codice e descrizione ed una tabella *T_ARTICOLO* con i campi codice e descrizione. Affinché Hibernate possa memorizzare trasparentemente istanze di *Articolo* nelle righe di *T_ARTICOLO* è necessario scrivere un descrittore che dica su per giù qualcosa del genere:

- le istanze di *Articolo* saranno memorizzate nella tabella *T_ARTICOLO*;
- l'attributo codice di *Articolo* sarà memorizzato nella colonna codice di *T_ARTICOLO;*
- l'attributo descrizione di *Articolo* sarà memorizzato nella colonna descrizione di *T_ARTICOLO.*

Come vedremo, il descrittore non sarà scritto proprio così, in italiano intendo, ma, come ci si potrebbe aspettare, in XML. Il descrittore gioca quindi un ruolo fondamentale: è il mediatore tra oggetti e tabelle, tra Java e il database sottostante e dipende fortemente da quest'ultimo. Cambiare il database significa necessariamente modificare il descrittore, poiché esso contiene tipi e notazioni specifiche per un particolare database. Come programmatori Java, e in generale come object oriented designer, ci siamo spesso scontrati con il problema di salvare il contenuto informativo di un oggetto in un database relazionale. Tale operazione, nota come object-relational mapping, è alquanto tediosa e può portare ad errori e bug che non dipendono dalla logica applicativa, bensì dal meccanismo di mapping. La ragione storica per cui mappare oggetti in un database relazionale è necessario, è dovuta alla diversa evoluzione dei linguaggi di programmazione rispetto a quella di database. I linguaggi object-oriented oggi la fanno da padrone; non solo Java e C++, ma anche Visual Basic, ASP, PHP e linguaggi di scripting adottano oramai estensioni orientate agli oggetti. Di conseguenza, oggi, si scrive e si progetta pensando in termini di classi, ereditarietà e polimorfismo. D'altra parte, i database più diffusi sono ancora quelli relazionali (nati prima dell'avvento della OOP), basati su un approccio notevolmente diverso da quello orientato agli oggetti, costituito da concetti quali *tabelle, chiavi primarie, chiavi esterne* e *Stored Procedure.* Prendendo atto di questo scenario, è evidente che trovare un modo per memorizzare oggetti in tabelle è un problema che affligge la quasi totalità dei programmatori che lavora nell'ambito dei database. Chi usa database object-oriented è immune al problema del mapping, ma tali database non sono tuttavia

PRIMA DI INIZIARE

Dopo aver scaricato Hibernate ed averlo installato in una directory del vostro hard-disk, per utilizzarlo nella vostra applicazione è necessario aggiornare il *CLASSPATH* del progetto in modo che includa le librerie di terze-parti presenti nella *Tabella 1*. Esse sono presenti nella directory *./lib,* la quale contiene anche altre librerie opzionali. Inoltre è necessario aggiungere nel *CLASSPATH* anche la libreria di Hibernate, rappresentata dal file *hibernate2.jar,* presente nella directory base del prodotto.

Utilizza questo spazio per le tue annotazioni

EJB CONTAINER-MANAGED PERSISTENCE

Un EJB di tipo CMP è un particolare entity Enterprise JavaBean in grado di "mapparsi" automaticamente con il database sottostante. Ciò avviene grazie a particolari descrittori scritti in XML che guidano l'operazione di mapping. Per maggiori dettagli consultare http://java.sun.com/developer/technicalArticles/ebeans/EJB20CMP.

così diffusi e sono spesso relegati a compiti specialistici. In Java esistono varie alternative, più o meno efficaci, per risolvere il problema. L'approccio più rudimentale è quello di usare JDBC per memorizzare il contenuto di un oggetto nella relativa tabella di un database relazionale. Tale operazione non richiede nessun tool aggiuntivo, ma è compito del programmatore scrivere il codice Java/SQL per memorizzare gli oggetti ed effettuare query. Non c'è nessuna trasparenza: per ogni classe è necessario implementare "a mano" del codice che si interfaccia col database. Una seconda strada, adottabile in contesti enterprise (J2EE), è quella di utilizzare particolari tipi di EJB denominati *CMP* (vedi riquadro). Sun ha poi lanciato una nuova API chiamata *Java Data Objects (JDO)*, nata con l'ambizione di diventare la soluzione definitiva al problema del mapping, ma a quanto pare non lo è, visto il diffondersi di altri tool di successo quali appunto Hibernate. Nel riquadro **Prodotti JDO** è presente una lunga lista dei tool più diffusi basati su *JDO*. Quindi Hibernate, che funziona più o meno come JDO, ma è più semplice e a mio avviso anche più efficace.

IL DATABASE

È fondamentale comunicare ad Hibernate quale database si intende utilizzare. Ciò può essere fatto impostando alcune proprietà nel file hibernate.properties. Nella directory *./etc* è presente il file *hibernate.properties* che può essere usato come modello. Esso mostra come configurare tutti i database supportati da Hibernate. Ovviamente voi utilizzerete solo le proprietà relative al vostro db. Se per esempio volete connettervi ad un database chiamato *"miodb"* di MySQL, allora il vostro file file *hibernate.properties* sarà il seguente:

```
## MySQL
hibernate.dialect net.sf.hibernate.dialect.MySQLDialect
hibernate.connection.driver_class com.mysql.jdbc.Driver
hibernate.connection.url jdbc:mysql://miohost.com/miodb
hibernate.connection.username pippo
hibernate.connection.password pluto
```

La prima riga indica la classe di Hibernate, chiama dialect, che si intende utilizzare per dialogare, nell'opportuno dialetto SQL, con il database. Nel nostro caso la classe dialect sarà quella relativa a MySQL. C'è una classe dialect per ogni database supportato e in **Tabella 2** si può vedere un elenco di tutti i database supportati da Hibernate e relative classe *dialect*. La seconda riga indica il driver JDBC da usare. È necessario che la libreria che implementa il driver sia anch'essa presente nel *CLASSPATH*. Nel caso specifico di MySQL 3 la libreria è rappresentata dal file *mysql-connector-java-3.0.7-stable-bin.jar*. La terza riga localizza il database all'interno di MySQL, esso si trova sulla macchina *"miohost"* e si chiama *"miodb"*. Le ultime due righe sono le informazioni relative all'utente che accede al database, ovvero username e password. Se invece usate Oracle, il vostro *hibernate.properties* potrebbe essere il seguente:

```
## Oracle
hibernate.dialect net.sf.hibernate.dialect.Oracle9Dialect
```

Libreria	Jar
CGLIB bytecode generator	*cglib-full.jar*
Apache Common Collections	*commons-collections-2.1.1.jar*
Apache Common Logging	*commons-logging-1.0.4.jar*
DOM4J	*dom4j-1.4.jar*
EHCache	*ehcache-0.9.jar*
Standard JDBC APIs	*jdbc2_0-stdext.jar*
Standard JTA API	*jta.jar*

TABELLA 1: Librerie di terze-parti obbligatorie per Hibernate.

CREIAMO UN NUOVO PROGETTO

1 Da eclipse scegliamo ***file/new project*** e poi ***Java Application.*** Nel passo che segue scegliamo un nome adeguato. Per il nosto esempio useremo "***store***"

SISTEMIAMO IL CLASSPATH

2 Da ***project/properties/java build path/add external jars*** includiamo i file necessari come in tabella uno. Aggiungiamo anche i ***jar*** del ***connector jdbc*** di MySQL

CREIAMO IL DATABASE

3 Abbiamo usato ***phpmyadmin*** per creare un database chiamato ***store***, con i campi citati nell'articolo e composti come in figura. Chiaramente potete usare qualunque altro metodo

```
hibernate.connection.driver_class
                        oracle.jdbc.driver.OracleDriver
hibernate.connection.username pippo
hibernate.connection.password pluto
hibernate.connection.url
                   jdbc:oracle:thin:@miohost:1521:miodb
```

IL PRIMO OGGETTO PERSISTENTE

Supponiamo di avere la classe *Article* così implementata:

```
public class Article {
  private String id;
  private String code;
  private String description;
  private double price;
  public String getId() {
    return id;}
  public void setId(String id) {
    this.id = id;}
  public String getCode() {
    return code;}
  public void setCode(String code) {
    this.code = code;}
  public String getDescription() {
    return description;}
  public void setDescription(String description) {
    this.description = description;}
  public double getPrice() {
    return price;}
  public void setPrice(double price) {
    this.price = price;} }
```

e di voler rendere persistenti le sue istanze in un database MySQL. Prima di tutto è necessario configurare il file hibernate.properties come mostrato nella sezione precedente. Poi, bisogna creare la tabella che ospiterà gli oggetti. Chiamiamo questa tabella *T_ARTICLE* ed avrà le colonne:

```
VARCHAR(100) id
VARCHAR(100) code
VARCHAR(100) description
DOUBLE       price
```

La differenza tra id e code è che mentre quest'ultimo è il codice dell'articolo, e fa parte della logica applicativa, id è un identificativo unico all'interno del database (e non solo all'interno degli articoli), generato dal sistema, per identificare un particolare oggetto indipendentemente dal suo significato applicativo. Questo id sarà utilizzato da Hibernate per identificare gli oggetti e, come vedremo nel prossimo articolo, metterli in relazione con altri. Adesso arriva la parte più interessante: scrivere il descrittore. Il nome del file deve essere uguale alla classe e deve avere l'estensione *hbm.xml*, quindi nel nostro caso *Article.hbm.xml*. Hibernate considererà tale file al pari di una risorsa o di un class file, quindi deve poter essere raggiungibile a runtime. Il file, scritto specificatamente per MySQL, sarà il seguente:

UUID

La sigla *UUID* sta per *Universal Unique IDentifier* e rappresenta un valore a 128 bit per identificare un oggetto in modo univoco.
Il sistema che lo genera garantisce l'univocità generando i bit da informazioni prelevate dall'hardware, ora corrente e numeri casuali.

Database	Classe Dialect
HypersonicSQL	*net.sf.hibernate.dialect.HSQLDialect*
PostgreSQL	*net.sf.hibernate.dialect.PostgreSQLDialect*
DB2	*net.sf.hibernate.dialect.DB2Dialect*
DB2/400	*net.sf.hibernate.dialect.DB2400Dialect*
MySQL	*net.sf.hibernate.dialect.MySQLDialect*
Oracle	*net.sf.hibernate.dialect.Oracle9Dialect* *net.sf.hibernate.dialect.OracleDialect*
Sybase	*net.sf.hibernate.dialect.SybaseDialect*
Mckoi SQL	*net.sf.hibernate.dialect.MckoiDialect*
SAP DB	*net.sf.hibernate.dialect.SAPDBDialect*
MS SQL Server	*net.sf.hibernate.dialect.SQLServerDialect*
Interbase	*net.sf.hibernate.dialect.InterbaseDialect*

TABELLA 2: Database supportati da Hibernate e relative classi dialect.

ISTRUIAMO HIBERNATE

```
hibernate.dialect net.sf.hibernate
                        .dialect.MySQLDialect
hibernate.connection.driver_class
                        org.gjt.mm.mysql.Driver
hibernate.connection.driver_class
                        com.mysql.jdbc.Driver
hibernate.connection.url
                        jdbc:mysql:///store
hibernate.connection.username jaco
hibernate.connection.password obfuscated
```

4 **Verificate che nel file *hibernate .properties* siano presenti le linee di informazione per la connessione. Sostituite *db, password* e *username* con valori appropriati**

CREIAMO IL FILE DI MAPPING

```
<hibernate-mapping>
  <class name="Article" table="T_ARTICLE">
    <id name="id" type="string"
                  unsaved-value="null">
      <column name="id" sql-type=
        "varchar(100)>" not-null="true" />
      <generator class="uuid.hex"/>
    </id>
[...]
    <property name="price">
      <column name="price" sql-type=
                "double" not-null="true"/>
    </property>
  </class>
```

5 **Utilizzate il codice come dichiarato nell'articolo e salvate tutto nel file *Article.hbm.xml* nella directory di esecuzione dei sorgenti, di modo che il path sia corretto**

ASSEMBLIAMO IL TUTTO

```
public class Article {
  private String id;
  private double price;
  public String getId() {
    return id;}
[...]
public static void main(String[] args) {
  Configuration cfg= new Configuration()
                  .addClass(Article.class);
[...]
```

6 **Create la classe *Article* e componente il file *store.java* come dichiarato nell'articolo. Lanciate tutto con > *java store* e controllate il risultato**

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-mapping
    PUBLIC "-//Hibernate/Hibernate Mapping DTD//EN"
    "http://hibernate.sourceforge.net/
                            hibernate-mapping-2.0.dtd">
<hibernate-mapping>
  <class name="Article" table="T_ARTICLE">
    <id name="id" type="string" unsaved-value="null" >
      <column name="id" sql-type="varchar(100)"
                              not-null="true"/>
      <generator class="uuid.hex"/>
    </id>
    <property name="code">
      <column name="code" sql-type="varchar(100)"
        unique="true" not-null="true"/>
    </property>
    <property name="description">
      <column name="description" sql-type="varchar(100)"
        not-null="true"/>
    </property>
    <property name="price">
      <column name="price" sql-type="double"
                              not-null="true"/>
    </property>
    </class>
</hibernate-mapping>
```

INSTALLARE IL CONNECTOR PER JDBC

Dalla directory JDBC presente nel CD di ioProgrammo semplicemente scompattare il file relativo. Per utilizzarlo è sufficiente inserirlo nel classpath.

	code	description	price	id
1	A-1001	iPod Mp3 Player	240.00	8af0829300a97b96
2				
3				

Fig. 1: ***La tabella T_ARTICLE dopo aver reso persistente un'istanza della classe Article***

Il documento aderisce al DTD presente in *hibernate.sourceforge.net/hibernate-mapping-2.0.dtd* e definisce come mappare l'oggetto *Article* nella tabella *T_ARTICLE*. Tralasciamo per il momento l'elemento id e dedichiamoci invece all'elemento *property*. Come si può notare c'è un elemento property per ogni attributo di *Article*. Molto semplicemente, esso abbina l'attributo della classe *Article* con la relativa colonna della tabella *T_ARTICLE*. L'elemento *column* specifica la colonna *target* e alcune caratteristiche di essa, come ad esempio il tipo SQL e vincoli vari, tipo *unique* e *not-null*. L'elemento *id* definisce invece l'identificativo unico per l'oggetto, che per la classe è l'attributo id e per la tabella la colonna *id*. L'elemento *generator* dice che l'*id* deve essere generato automaticamente usando la specifica *uuid* (vedi riquadro). Questo è tutto: abbiamo Hibernate, abbiamo un database, una classe di cui memorizzare oggetti e il descrittore; siamo pronti per rendere permanente il nostro primo oggetto. La classe che effettuerà quest'operazione si chiamerà Store; per semplicità tutto il codice sarà implementato nel metodo *main*. I package di Hibernate da importare sono:

import net.sf.hibernate.;*
import net.sf.hibernate.cfg.;*

Le prime istruzioni del metodo *main* saranno le seguenti:

```java
public class Store {
  public static void main(String[] args) throws Exception {
    Configuration cfg =
                  new Configuration().addClass(Article.class);
    SessionFactory sf = cfg.buildSessionFactory();
...
```

Come si può notare si crea un oggetto di tipo *Configuration, cfg,* e si invoca su di esso il metodo *addClass* passando *Article.class*. Ciò informerà Hibernate che le istanze della classe *Article* possono essere rese persistenti. A questo punto si crea una *SessionFactory* che ci sarà utile successivamente per creare una sessione Hibernate. Una sessione rappresenta un ciclo di vita nella quale si può operare sul database mediante Hibernate. Quindi possiamo creare il nostro oggetto *Article*:

```java
Article article = new Article();
article.setCode("A-1001");
article.setDescription("iPod Mp3 Player");
article.setPrice(240.00);
```

Infine, è necessario creare una sessione e far partire una transazione al fine di salvare l'oggetto:

```java
Session session = sf.openSession();
Transaction t = session.beginTransaction();
// Salva l'oggetto
session.save(article);
t.commit();
session.close();
```

Avremo modo di approfondire concetti quali sessioni e transazioni nei prossimi articoli di questa serie. In ogni caso, credo che l'operazione di "salvataggio" sia molto semplice ed intuitiva. Eseguendo il programma, > *java Store;* renderemo persistente l'articolo, inserendolo nel database. Per essere certi che l'operazione sia andata a buon fine, basta controllare che la tabella *T_ARTICOLO* contenga la riga relativa all'articolo *"A-1001"*, come mostrato in **Figura 1**.

Hibernate -
http://www.hibernate.org

Davor Cengija, Hibernate Your Data,
http://www.onjava.com/pub/a/onjava/2004/01/14/hibernate.html

CONCLUSIONI

In questo primo articolo su Hibernate abbiamo parlato di *mapping O/R* e come può essere realizzato da Hibernate mediante i descrittori. Inoltre, abbiamo visto come configurare ed utilizzare il prodotto con MySQL e rendere persistenti oggetti relativi ad una classe Java. Nel prossimo numero vedremo come attuare relazioni tra classi e mapparle in opportune tabelle del database.

Giuseppe Naccarato

La guida definitiva per comprendere a fondo ADO

Accesso ai DB con Visual Basic .Net 2003

Parliamo di tutte le tecniche fondamentali per interagire con i database, utilizzando gli strumenti di sviluppo proposti da Visual Basic .Net 2003

La tecnologia ADO.NET definisce il modello di programmazione ad oggetti per accedere ad una fonte dati e mantenerla aggiornata. Utilizzando ADO.NET è possibile estrarre dati da Microsoft SQL Server o da sorgenti dati OLE DB più generiche, elaborarli ed aggiornare le tabelle originali del database. In questa serie di articoli, descriveremo brevemente il generico modello ad oggetti di ADO.NET e, successivamente, ci soffermeremo sull'impiego di ADO .NET per accedere ad un database Sql Server 2000. Daremo, inoltre, uno sguardo all'interfaccia di Sql Server 2000

I PROVIDER DI DATI

I data provider di VB.NET hanno funzione di ponte tra l'applicazione e la sorgente dati.
Descrivono un insieme di classi che consentono alle applicazioni di leggere e scrivere dati memorizzati in un database.
Offrono, nel loro insieme, un accesso veloce ed affidabile ad un'ampia gamma di origini dati oltre che a specifici database come Sql Server, Access ed Oracle. Visual Studio .Net 2003, fornisce i seguenti provider:

- Il provider di dati .NET *SQL Server*, ottimizzato per SQL Server 7.0, ed SQL Server 2000.
- Il provider di dati .NET *OLE DB,* che permette di accedere ad una sorgente dati per la quale esiste un provider OLE DB. È il provider indicato per accedere a Microsoft Access basato sul motore Jet 4.0.
- Il provider di dati .NET *ODBC,* che permette di accedere ad una sorgente dati per la quale esista un driver ODBC.
- Il provider di dati .NET *Oracle,* che permette di accedere ad origini dati Oracle tramite software di connessione per client Oracle.

ADO.NET, IL MODELLO AD OGGETTI

Qualunque sia il Data Provider che dovremo utilizzare, si avranno sempre a disposizione quattro classi generiche (che non dovremo utilizzare nel codice con i nomi indicati di seguito):

- **Connection** - *Stabilisce una connessione ad un'origine dati specifica.* La funzione dell'oggetto *Connection* è quella di stabilire una connessione alla fonte dati, permettendo di aprire e chiudere una connessione ad un database. Espone la proprietà *ConnectionString* che permette di definire, tra l'altro, il nome del database d'origine, i metodi *Open* e *Close* per aprire e chiudere la connessione ed il metodo *BeginTransaction* per avviare una transazione.

- **Command** - *Permette di eseguire un comando su un'origine dati.* La funzione dell'oggetto *Command* è quella di consentire l'esecuzione di istruzioni SQL su un database. È possibile eseguire query di interrogazione, inviare un qualsiasi comando sql (come un'istruzione di *insert* o di *update*), oppure invocare una stored procedure. Tutte queste operazioni sono possibili grazie ai vari metodi *Execute.*

- **DataReader** - *Permette di leggere un flusso di dati forward-only in sola lettura, proveniente da un'origine dati specifica.* Il *DataReader* è utilizzato insieme all'oggetto *Command* e viene istanziato dopo una chiamata al metodo *ExecuteReader*. Mentre è aperto l'oggetto *DataReader* che utilizza un oggetto *Connection,* non è possibile utilizzare lo stesso oggetto *Connection* per nessun altro scopo se non chiudere la connessione.

- **DataAdapter** - *Consente la comunicazione tra una fonte dati ed un oggetto DataSet e risolve gli*

In VB.NET è ancora possibile utilizzare gli oggetti della precedente tecnologia ADO utilizzata per l'accesso ai dati in VB6. È sufficiente selezionare la voce di menu *Progetto/Aggiungi Riferimento* e, nella finestra di dialogo, scegliere la componente *adodb* dalla scheda .NET.

aggiornamenti con l'origine dati. Il *DataSet* rappresenta l'oggetto principale dell'architettura disconnessa di ADO .NET. È paragonabile ad un database relazionale di piccole dimensioni memorizzato sul client e non dipende da alcun database specifico. Espone una collezione di oggetti *DataTable*, ognuno dei quali contiene un set di risultati, tipicamente popolato da una query, su una tabella del database. Un oggetto *DataTable* è costituito, a sua volta, da una collezione di oggetti *DataRow*, dove ciascuno di tali oggetti contiene un diverso record, risultato dalla query di selezione. Il *DataSet* contiene, inoltre, una collezione di oggetti *DataRelation*, ognuno dei quali corrisponde ad una relazione tra oggetti *DataTable* differenti, in pratica come le relazioni che intercorrono tra le tabelle di un database relazionale.
L'oggetto *DataAdapter* costituisce il ponte tra l'oggetto *Connection* e il *DataSet*. Con il suo metodo *Fill* si popola il *DataSet*, mentre con il metodo *Update* si aggiorna il database con i record modificati nel *DataSet*.

Il nome di questi oggetti, da utilizzare nel codice, dipende dal *Data Provider* che dovremo utilizzare, vediamoli in dettaglio.

I NAMESPACE DI ADO.NET

Il Framework .NET è un'ampia raccolta di classi suddivise in un vasto insieme di namespace (spazi dei nomi), che raggruppano le classi simili. La maggior parte delle classi del Framework è riunita in un namespace denominato *System*, in particolare le classi cui fa riferimento ADO .NET sono raggruppate nei seguenti namespace:

- Il namespace **System.Data** contiene gli oggetti di ADO.NET che non fanno parte di uno specifico data provider. Ad esempio, questo namespace contiene l'oggetto *DataSet* e tutti i relativi oggetti secondari, come *DataTable, DataColumn, DataRow* e *DataRelation.*

- Il namespace **System.Data.Common** contiene le classi condivise dai provider di dati .NET come gli oggetti *DataAdapter* ed altre classi vir-

CREAZIONE DI UN DATABASE SQL SERVER 2000

Ecco i passi necessari per creare un nuovo database in SQL Server 2000

1 **La prima operazione da compiere è quella di espandere il ramo del Server fino alla cartella Database. Le possiamo fare con gli strumenti di gestione classica**

2 **Clicchiamo con il tasto destro del mouse sulla cartella Database, e dal menù contestuale selezioniamo la voce: Nuovo Database. In questo modo si aprirà la finestra di dialogo Proprietà Database.**

3 **La finestra di dialogo Proprietà Database è costituita da tre schede. Nella scheda Generale dobbiamo indicare il nome logico del nuovo database ad esempio: MiaRubrica**

4 **Nella scheda File di dati possiamo cambiare il nome del file fisico del database e la sua posizione su disco, con valori diversi da quelli proposti per default (per il nostro esempio lasciamo quelli proposti). Possiamo, inoltre, cambiare la dimensione iniziale (ad es. 10 MB), ed attivare l'opzione di crescita automatica delle dimensioni del database. In generale è consigliabile impostare un aumento in termini percentuali, se poi non ci sono particolari problemi di spazio, possiamo attivare l'opzione aumento illimitato in modo da non porre alcun limite alla crescita del database.**

5 **Nella scheda Log delle transazioni possiamo modificare le proprietà del file che conterrà la cronologia delle transazioni effettuate sul database, il file di Log.**

tuali. Queste classi sono utilizzate come classi di base per diversi oggetti appartenenti ai namespace descritti di seguito.

- Il namespace **System.Data.OleDb** contiene le classi che costituiscono il provider di dati .NET Ole Db, come *OleDbConnection, OleDbCommand, OleDbDataReader* e *OleDbDataAdapter.*
- Il namespace **System.Data.SqlClient** contiene le classi utilizzate per accedere ad un'origine dati SQL Server, come *SqlConnection, SqlCommand, SqlDataReader* e *SqlDataAdapter.*
- Il namespace **System.Data.Odbc** contiene le classi utilizzate per accedere ad un'origine dati ODBC, come *OdbcConnection, OdbcCommand, OdbcDataReader* e *OdbcDataAdapter.*
- Il namespace **System.Data.OracleClient** contiene le classi che permettono di accedere ad un'origine dati Oracle, come *OracleConnection, OracleCommand, OracleDataReader* e *OracleDataAdapter.*

Per accedere ad Sql Server 2000, utilizzeremo i namespace: *System.Data.SqlClient* e *System.Data*. Per utilizzare, all'interno del codice, un oggetto contenuto in uno di questi namespace, si dovrà fare riferimento all'oggetto, facendolo precedere dal nome del namespace che lo contiene.
Ad esempio, per utilizzare un oggetto *SqlConnection*, si dovrà scrivere:

```
System.Data.SqlClient.SqlConnection()
```

Per evitare di scrivere sempre il nome del namespace, si può assumere di avere utilizzato le istruzioni *Imports* riportate nel box a lato. Prima di continuare con Ado .Net uno sguardo all'interfaccia di Sql Server 2000

SQL SERVER ENTERPRISE MANAGER

Le principali funzionalità dell'ambiente di sviluppo di SQL Server sono accessibili mediante *Enterprise Manager*. Potremmo definire *Enterprise Manager* come la centrale di controllo di Microsoft SQL Server, esso offre tutti gli strumenti per eseguire e monitorare le funzioni vitali di SQL Server.
Come si può osservare, in **Figura 1**, l'interfaccia di *Enterprise Manager* è organizzata in tre sezioni:

- *Barra degli strumenti e dei menù* (in alto)
- *Struttura ad albero della consolle* (a sinistra)
- *Pannello di visualizzazione dettagli* (a destra)

Al livello gerarchicamente più alto della struttura troviamo l'icona Microsoft SQL Server. Questo nodo raccoglie in se tutte le istanze di SQL Server che sono attualmente registrate. Le istanze registrate sono rappresentate da un icona del tipo seguente dove il quadrato rosso ed il triangolo verde indicano rispettivamente lo stato di arresto o di esecuzione dell'istanza di SQL a cui si riferisce. Un istanza SQL viene identificata univocamente dalla macchina su cui risiede fisicamente e dal sistema operativo della macchina stessa. All'interno della cartella Database sono elencati tutti i database presenti sul Server selezionato. Per ciascun database sono riportati una serie di oggetti quali *Tabelle, Stored Procedure, Diagrammi, Utenti,* ecc... I contenuti delle altre cartelle sono riportati nel box a lato.

L'OGGETTO SQLCONNECTION

La prima azione da eseguire quando si vuole utilizzare una fonte dati è quella di aprire una connessione verso quest'ultima, ciò significa creare un oggetto *SqlConnection*, impostare la stringa di connessione ed aprire la connessione tramite il metodo *Open*. Per operare in modalità connessa, i punti appena elencati si devono eseguire soltanto alla partenza dell'applicazione, mentre alla fine dell'applicazione si deve chiudere la connessione tramite il metodo *Close*. Per operare in modalità disconnessa, invece, i punti precedenti devono essere eseguiti ogni volta che si deve accedere al database.
Riassumendo si deve:

- Creare un oggetto *SqlConnection.*
- Impostare la stringa di connessione ad uno specifico database.
- Aprire la connessione tramite il metodo *Open.*
- Eseguire le operazioni sul database con uno degli oggetti Ado .Net che analizzeremo in seguito.
- Chiudere la connessione tramite *Close.*

Per creare un oggetto SqlConnection si deve utilizzare la solita sintassi usata per creare un qualsiasi oggetto:

```
Dim ObjConnection As New SqlConnection()
```

Fig. 1: L'interfaccia di Enterprise Manager

NOTA

Per mantenere il codice il più compatto possibile, si può utilizzare l'istruzione *Imports* che semplifica l'accesso alle classi, eliminando la necessità di digitare in modo esplicito il nome completo del namespace che le contiene. Le istruzioni *Imports* devono sempre essere scritte nella parte superiore del file nel quale si vogliono utilizzare, prima di qualunque altro codice. Per queste ragioni, in tutti gli esempi di codice, è ragionevole assumere l'inserimento delle seguenti istruzioni *Imports:*

```
Imports System.Data
Imports System.Data
    .SqlClient
```

LA PROPRIETÀ CONNECTIONSTRING

La proprietà principale dell'oggetto SqlConnection è la proprietà *ConnectionString* che permette di impostare la stringa di connessione al database. La stringa di connessione consente di definire il nome del database di origine ed altri parametri necessari a stabilire la connessione iniziale, delimitati da punto e virgola. I parametri che vengono settati sono:

- L'attributo **Provider**, che determina il nome del provider OLE DB da utilizzare per connettersi ai dati. Nel caso dell'oggetto *SqlConnection*, questo attributo non è necessario poichè il Data Provider SQL Server consente di connettersi esclusivamente ad un database SQL Server.
- L'attributo **Data Source**, che determina il nome del computer sul quale è installato il database, oppure il path di installazione di un database Access. È possibile usare come attributo localhost nel caso dobbiamo connetterci ad SQL Server sulla macchina locale.
- L'attributo **User ID** che specifica lo username dell'utente che esegue il login.
- L'attributo **Password**, che specifica la password dell'utente che esegue il login.
- L'attributo **Initial Catalog**, che specifica il nome del database da utilizzare.
- L'attributo **Integrated Security**, che indica se si deve utilizzare la protezione integrata per effettuare una connessione affidabile a SQL Server. Specificando il valore *SSPI (Security Support Provider Interface)* non è necessario utilizzare il nome utente e la password nella stringa di connessione, delegando al sistema operativo la gestione della sicurezza.

Per il nostro caso si potrà quindi scrivere:

```
ObjConnection.ConnectionString = "Initial
                        Catalog=MiaRubrica;Data
 Source=localhost;Integrated Security=SSPI;"
```

CREAZIONE DI UNA TABELLA

Una volta creato il nostro Database dobbiamo definire le entità fondamentali che permetteranno di utilizzarlo: le tabelle

1 La prima operazione da compiere è quella di espandere la cartella *Database*, selezionare il Database di esempio (*MiaRubrica*) e cliccare con il tasto destro del mouse. Dal menù contestuale possiamo selezionare il menu: *Nuova Tabella*

2 Nella finestra di dialogo *Nuova Tabella* è possibile definire le colonne, il tipo di dati, specificarne le chiavi primarie ecc. Si aprirà la finestra di dialogo: *Nuova Tabella*

3 Per definire una colonna, dobbiamo cliccare nel campo *Nome Colonna* e digitare il nome del campo (*CodicePersona* nell'esempio). La tabella sarà automaticamente modificata con il nuovo campo

4 Cliccando nel campo *Tipo Dati* il sistema mostra un menù con tutti i tipi di dati disponibili, da cui possiamo selezionare il tipo desiderato (ad esempio *int*).

5 In base al tipo selezionato, il campo della colonna *Lunghezza* sarà editabile o meno e consentirà di specificare la lunghezza, o la lunghezza massima del campo (nel caso del tipo *int* non è possibile cambiarne la dimensione).

6 Come indicato nei passi precedenti, possiamo completare la definizione dei campi. Ad esempio definiamo i campi: *Nome, Cognome* e *NumeroTelefono*

APRIRE E CHIUDERE LA CONNESSIONE

Dopo aver istanziato un oggetto *SqlConnection*, con la opportuna stringa di connessione, è necessario utilizzare il metodo *Open* per collegarsi alla fonte dati:

```
ObjConnection.Open()
```

Per liberare le risorse di un oggetto *SqlConnection* quando non sono più necessarie, si deve utilizzare il metodo *Close*

```
ObjConnection.Close()
```

Il metodo *Close* esegue il rollback di tutte le transazioni in sospeso e rilascia la connessione, se viene chiamato quando non ci sono connessioni aperte non viene generata alcuna eccezione. Come si può osservare dal codice di esempio, i metodi *Open* e *Close* non accettano argomenti. Quando si opera in modalità disconnessa diventa molto importante gestire gli errori, per evitare di lasciare connessioni inattive aperte, saturando il sistema. A causa del meccanismo di garbage collection di VB .NET, quando l'oggetto *SqlConnection* esce dal proprio ambito di visibilità la connessione non viene chiusa automa-ticamente ma potrebbe essere chiusa diverso tempo dopo. Per questo è importante chiudere esplicitamente l'oggetto *SqlConnection*.

7 **Dopo aver completato la definizione dei campi, possiamo impostare la chiave primaria della tabella selezionando il campo (*CodicePersona*) e cliccando sull'icona a forma di chiave sulla barra degli strumenti**

8 **Infine dobbiamo salvare la tabella appena creata, cliccando sull'icona a forma di dischetto sulla barra degli strumenti e definendo il nome della tabella nella finestra di dialogo *Nome Tabella* (ad Es. *Persona*)**

L'OGGETTO SQLCOMMAND

Dopo aver aperto una connessione, siamo pronti per interagire con il database per mezzo dell'oggetto *SqlCommand*. L'oggetto *SqlCommand* deve essere associato ad un oggetto *SqlConnection*, preventivamente connesso alla sorgente dati, e può contenere una query di selezione (per leggere i dati dal database) o una query d'azione (per aggiornare i dati), impostata tramite la proprietà *CommandText*. Dopo aver impostato l'oggetto *SqlCommand*, si esegue, quindi, uno dei relativi metodi *Execute:*

- **ExecuteNonQuery** si utilizza per inviare la query d'azione specificata da *CommandText* al database, e restituisce il numero di record coinvolti. Permette attività di manipolazione di dati, come inserimenti, aggiornamenti e cancellazioni corrispondenti alle istruzioni SQL di *Insert, Update* e *Delete.*

- **ExecuteReader** si utilizza per inviare la query di selezione specificata da *CommandText* al database, e restituisce l'oggetto *DataReader* che consente di accedere al set dei risultati *(resultset).*

- **ExecuteScalar** si utilizza per inviare la query di selezione specificata da *CommandText* al database, e restituisce un singolo valore risultato di un'istruzione *Select* (valore scalare). Il metodo restituisce la prima colonna, della prima riga di un gruppo di risultati, ignorando tutti gli altri valori. È consigliabile usare questo metodo, ad esempio, se il risultato è il frutto di query con clausole di aggregazione come *count* o *sum.*

NOTA

GESTIRE LE TRANSAZIONI

La classe *Connection* di ADO espone i metodi *BeginTrans, CommitTrans* e *RollbackTrans,* che consentono, rispettivamente, di avviare, effettuare il commit o annullare una transazione. L'oggetto *SqlConnection* di ADO.NET espone soltanto il metodo *BeginTransaction* che restituisce un oggetto SqlTransaction. Si utilizza, quindi, l'oggetto *SqlTransaction* per controllare il risultato della transazione: si invoca il metodo *Commit* per confermare tutte le modifiche ed il metodo *Rollback* per annullarle.

CONCLUSIONI

In questo primo articolo abbiamo analizzato il generico modello ad oggetti di Ado .Net soffermandoci sulle classi che ci permetteranno di accedere ad un database Sql Server 2000.
Nel prossimo articolo introdurremo alcuni esempi pratici che ci aiuteranno a manipolare i dati del database MiaRubrica.

Luigi Buono

Dotiamo gli script Python di interfacce grafiche portabili

Python power La nuova era!

Impariamo come programmare interfacce grafiche in Python. Appena un accenno delle incredibili potenzialità di questo linguaggio, che merita di essere usato e diffuso

Utilizza questo spazio per le tue annotazioni

Forse non tutti i lettori hanno avuto il piacere di programmare in Python, quindi è opportuno descriverne brevemente le caratteristiche salienti. I linguaggi di scripting consentono, per natura, di ridurre notevolmente le righe di codice necessarie per risolvere determinate classi di problemi. Python, essendo un linguaggio interpretato dal design curato e moderno, risulta: chiaro, compatto, elegante, versatile, portabile, modulare, estensibile, orientato agli oggetti e immediato da apprendere. La sua semplicità ne ha favorito la diffusione nei contesti più svariati: implementazione rapida di strumenti di supporto, integrazione di sistemi eterogenei, elaborazione del testo, programmazione Web e lato server, accesso generico a basi di dati, etc. Anche dal punto di vista economico Python offre diversi vantaggi: è gratuito ed open-source, la presenza di una ricca libreria standard riduce i tempi ed i costi di sviluppo, la chiarezza del codice agevola la manutenzione dei progetti. L'interprete Python è scritto in ANSI C, sono disponibili porting più o meno ufficiali per molte piattaforme: Windows, UNIX/Linux, Macintosh, Solaris, .NET Framework, PlayStation2, Amiga, AS/400, OS/2, Palm OS… Il nuovo approccio a Python sarà completamente pratico. Il linguaggio è talmente interessante nella sua curva di apprendimento che scopriremo tutti i suoi segreti usandolo a fini pratici.

LE ALTERNATIVE

In questo paragrafo analizzeremo tre librerie utilizzabili per creare interfacce grafiche in Python e sceglieremo la più adatta ai nostri scopi. Le aspiranti al titolo sono:

- **PyWin32** – Se il vostro progetto deve essere eseguito solo in ambienti Windows allora potete ricorrere a questa soluzione. *PyWin32* incapsula le funzioni dell'API Win32 e MFC ma pregiudica irrimediabilmente la portabilità del codice. *PyWin32* rende Python un linguaggio idoneo per scrivere ogni tipo di applicazione Windows: *GUI (Graphical User Interface)*, servizi, programmi basati su OpenGL o ActiveX, etc.
- **Tkinter** – Tutte le distribuzioni di Python includono il modulo *Tkinter* quindi non c'è bisogno di installare pacchetti aggiuntivi. Al contrario di *PyWin32* consente di creare programmi multipiattaforma anche se l'aspetto dei controlli non è sempre gradevole. *Tkinter* è un prodotto maturo, ma può risultare troppo semplicistico soprattutto nella gestione di interfacce complesse.
- **wxPython** – È una libreria emergente e performante se confrontata con *Tkinter*, poggiando sulle classi C++ che compongono *wxWidgets*

COME INIZIARE

TUTTO OK?

1 **Accertatevi che l'installazione sia andata a buon fine eseguendo un semplice doppio-clic su una qualsiasi applicazione demo di *WxPython*. Provate l'impressionante *Main.py*.**

(nota anche come *wxWindows*) garantisce un ottimo supporto ed una buona scalabilità. Inoltre, una quantità cospicua di controlli visuali è supportata in Windows, sistemi Unix-like e parzialmente in Mac. Si noti che il look-and-feel nativo è ottenuto grazie alla sottostante libreria *wxWidgets*.

La nostra scelta cade sulla terza opzione, *wxPython* oltre ad essere multipiattaforma e facile da usare dispone di un insieme completo di componenti. Forse è leggermente meno stabile rispetto alle librerie alternative ma sta maturando rapidamente. I prototipi realizzati mediante *wxPython* sono, nel caso in cui sorga la necessità, portabili in C++ senza eccessive complicazioni. Esistono astrazioni basate su *wxPython* che semplificano la vita del programmatore, a titolo di esempio cito *PythonCard*.

INTRODUZIONE A WXPYTHON

Come accennato in precedenza *wxPython* costituisce un *wrapper*, ossia un involucro, per le classi *wxWidgets* sviluppate in C++. Il programmatore può dunque istanziare oggetti *wxWidgets* e richiamare i relativi metodi direttamente in Python, sarà compito della libreria gestire nel modo corretto le associazioni. Non a caso la documentazione di *wxPython* ricalca in gran parte quella distribuita con *wxWidgets*, solo di rado si rendono necessarie delle annotazioni per evidenziare le differenze. Prima di proseguire installiamo i pacchetti *Python* e *wxPython* seguendo i passi descritti nel riquadro **Come iniziare**. Al fine di evitare incompatibilità è importante installare la versione di *wxPython* specifica per una data distribuzione di Python. La versione corrente per Windows è contenuta nel file denominato *Wxpython2.5-Win32-Ansi-2.5.3.1-Py23.exe* e, come si evince dal suffisso *Py23*, risulta compatibile con *Python 2.3.x*. Vi faccio notare che gli script Python possono essere creati inserendo le istruzioni in modo interattivo nell'interprete *python.exe* (**Figura 1**) oppure, a seconda delle proprie inclinazioni e disponibilità, mediante ambienti di sviluppo gratuiti o commerciali. Il nostro primo esperimento consiste nel creare una finestra vuota inserendo il codice direttamente nell'interprete o sfruttando lo script *idle.py* contenuto nella cartella *\lib\idlelib* delle distribuzioni ufficiali di Python.
L'espressività del linguaggio permette di sviluppare un programma completo scrivendo solo una manciata di righe di codice:

```
# Importa il namespace wx
import wx
# Crea un'applicazione wxPython
app = wx.PySimpleApp()
# Poi crea una finestra normale
finestra = wx.Frame(None, -1, "IoProgrammo!")
# Mostra la finestra appena creata
finestra.Show(1)
# Interagisce con l'utente ed il sistema operativo
app.MainLoop()
```

Come potete osservare dalla (**Figura 1**) la finestra assumerà l'aspetto nativo del sistema operativo in cui

***Fig. 1:** Il sorgente del nostro primo programma ed il risultato dell'esecuzione*

SELF
**A parte qualche differenza è l'equivalente dell'operatore *this* usato in C++/Java/C# per fare riferimento all'istanza corrente di una classe.
I programmatori Delphi conoscono un omonimo operatore presente in Object Pascal.**

PRIMA PYTHON

2 **Il primo pacchetto da installare è *Python-2.3.x*. La scelta della cartella è fondamentale perché, per evitare inutili complicazioni, dovrà coincidere con quella del passo successivo.**

POI WXPYTHON

3 **La procedura di installazione di *WxPython* rileva automaticamente la cartella in cui è stato installato Python, vi suggerisco di confermare il percorso proposto come in figura.**

DEMO E DOCUMENTAZIONE

4 ***Wxpython2.5-Win32-Docs-Demos-2.x.y.z.exe* può essere installato in una cartella qualsiasi, contiene il codice sorgente delle applicazioni di esempio e la documentazione ufficiale di *WxPython*.**

MONTY PYTHON

Il nome del linguaggio è legato alla passione del suo progettista, Guido Van Rossum, per la serie televisiva "Monty Python's Flying Circus".

ESTENSIONI & CO

C/C++ e Delphi sono solo alcuni dei linguaggi impiegabili nello sviluppo di estensioni per Python. Ricorrendo a strumenti specifici (es. SWIG) si possono integrare script Python nelle proprie applicazioni C/C++/Delphi/etc.

Fig. 2: ***Komodo: un ambiente di sviluppo commerciale per Python***

lo script viene eseguito, in questo caso Windows XP. I commenti al codice sono preceduti dal simbolo #. La prima riga è fondamentale in quanto serve ad importare ogni elemento del pacchetto *wx*, anche se in realtà potremmo limitarci all'importazione di un sottoinsieme di funzioni e costanti riducendo così il tempo di esecuzione. Nel primo esempio, a differenza dei successivi, non emerge in modo evidente la natura orientata agli oggetti di Python.

Fig. 3: ***Abbiamo aggiunto un pulsante alla finestra***

wx.Window è la classe base di tutti i controlli visuali come pulsanti, caselle di testo e cosi via, per i nostri scopi è invece necessario ereditare da *wx.Frame*. Nella dichiarazione della classe Applicazione creiamo un'istanza di *Finestra*, ovvero una classe derivata da *wx.Frame* con la minima aggiunta di un pulsante (**Figura 3**):

```
import wx
class Finestra(wx.Frame):
    def __init__(self):
        wx.Frame.__init__(self, None, -1, "ioProgrammo
                                                  con classe!")
        pulsante = wx.Button(self, -1, "Sono un pulsante
                                         inutile... per il momento!")
class Applicazione(wx.App):
     def OnInit(self):
         f = Finestra()
         f.Show(1)
         return 42
app = Applicazione()
# Loop di gestione dell'applicazione
app.MainLoop()
```

Sul cd allegato alla rivista trovate gli esempi proposti sotto forma di script con estensione *.py*, se l'installazione di Python è andata a buon fine sarà sufficiente un semplice doppio-clic per eseguirli. Gli amanti della linea di comando possono digitare il nome dello script, includendo l'estensione *.py*, seguito dalla pressione del tasto invio.

OLTRE LE BASI

Spero abbiate apprezzato la sinteticità del codice proposto finora. È ovvio che per creare un'applicazione "seria" significa: inserire e coordinare una certa quantità di controlli visuali, gestirne gli eventi, conoscere in modo dettagliato le classi presenti in *WxPython*. Come vedremo in seguito, possiamo avvalerci di strumenti *RAD (Rapid Application Development)* per semplificare la creazione di un'interfaccia grafica compatibile con *wxPython*. Per il momento poniamoci l'obiettivo di realizzare un visualizzatore di pagine HTML, impareremo a posizionare i controlli ed a rispondere agli eventi generati dall'utente. Una finestra può essere concepita partendo da una delle seguenti classi: *wx.Frame, wx.MiniFrame, wx.Dialog, wx.MDIParentFrame* o *wx.MDIChildFrame*. Se non avete bisogno di finestre modali di tipo *wx.Dialog* o interfacce a documenti multipli, in stile Microsoft Word per intenderci, la soluzione migliore è rappresentata dalla classe *wx.Frame*. Sono previsti diversi metodi per il posizionamento dei controlli in una finestra:

- **Metodo manuale** – la posizione e la dimensione dei singoli controlli è determinata dal programmatore e deve essere aggiornata ad ogni evento di ridimensionamento della finestra.
- **Vincoli di Layout** – Una tecnica sicuramente potente ma di non immediata attuazione, è riservata ai programmatori esperti, si consulti la documentazione della classe *wx.LayoutConstraints.*
- **Sizers** – Rappresentano un buon compromesso tra potenza e semplicità. Ricordano i LayoutManager di Java e la gestione dei layout nei toolkit

Fig. 4: ***L'editor di risorse XRCed, esporta in formato XML***

Qt e *GTK*. Il ridimensionamento della finestra non causa problemi perché è controllato automaticamente da *wxPython*. Dopo aver istanziato un *wx.BoxSizer* possiamo aggiungere controlli richiamando il metodo *Add* o la variante *AddMany*. Non ci sono troppe limitazioni sui sizer annidati. La maggior parte degli strumenti per la creazione assistita delle interfacce impiega esclusivamente i *sizer*. Uno di essi, *XRCed*, è mostrato in (**Figura 4**) e converte il layout prodotto dal programmatore in un file XML.

Fig. 5: ***Con poche righe di codice abbiamo disposto vari controlli***

A titolo di esempio la (**Figura 5**) è generata dal codice:

```
from wxPython.wx import *
...omissis...
box = wxBoxSizer(wxVERTICAL)
# Aggiungiamo diversi controlli in un colpo solo!
box.AddMany([(wxButton(self, -1 , "Ciao"),0, wxEXPAND),
  (wxButton(self, -1 , "da"),0, wxEXPAND),
  (wxButton(self, -1 , "IoProgrammo"),0, wxEXPAND)])
# Secondo wxBoxSizer
box1 = wxBoxSizer(wxHORIZONTAL)
box1.Add(wxButton(self, -1 , "SAL"), 0, wxEXPAND)
# Casella combinata
box1.Add(wxComboBox(self, -1, "Salvatore", (-1, -1),
(100, -1), "Salvatore", wxCB_DROPDOWN), 0,
                                                wxEXPAND)
# Orologio analogico
box1.Add(ac.AnalogClockWindow(self, style=
                      wxSUNKEN_BORDER), 1, wxEXPAND)
box.Add(box1, 1, wxEXPAND)
# Imposta il sizer per il frame
self.SetSizer(box)
```

La sezione *"Window Layout"* dell'applicazione dimostrativa *Main.py* include esempi sulle tecniche di posizionamento non descritte nell'articolo. Vale la pena di provarle per comprenderne le differenze e le potenzialità! Un programma non è un'entità statica, infatti interagendo con l'utente e con il sistema operativo è sottoposto a particolari "sollecitazioni" dette eventi. La pressione di un pulsante, il comando di chiusura o di ridimensionamento, la selezione di un elemento da una lista rappresentano solo alcuni esempi di eventi. Vediamo quale è il principale meccanismo previsto da *wxPython* per rispondere, durante l'esecuzione del ciclo *Mainloop*, a tali richieste. Il vecchio metodo basato sugli identificatori ottenuti per mezzo della funzione *wxNewId()* è stato affiancato da un nuovo meccanismo: il collegamento con il gestore di evento attraverso il metodo *Bind*. Un tempo era necessario associare un identificatore ad un controllo per poter richiamare la funzione di risposta ad un evento specifico:

```
from wxPython.wx import *
...omissis...
# Crea un identificatore
ID_VECCHIO = wxNewId()
# Associa l'identificatore ID_VECCHIO al pulsante1
pulsante1 = wxButton(self, ID_VECCHIO, "Vecchio")
# All'evento "pressione" è associato il gestore OnVecchio
EVT_BUTTON(self, ID_VECCHIO, self.OnVecchio)
# Se l'utente preme il pulsante viene mostrato un
                                               messaggio
def OnVecchio(self, event):
  wxMessageBox("Hai premuto il tasto VECCHIO")
```

Il vecchio meccanismo è laborioso e non scalabile quindi si consiglia di utilizzare il collegamento *controllo => gestore* di evento mediante Bind (**Figura 6**):

Fig. 6: ***Il metodo OnNuovo risponde alla pressione del pulsante "Nuovo"***

```
# Creo il pulsante
pulsante2 = wxButton(self, 1001, "Nuovo")
# Lego l'evento "pressione" al gestore OnNuovo
self.Bind(EVT_BUTTON, self.OnNuovo, pulsante2)
# Il metodo richiamato al momento della pressione di
                                               pulsante2
def OnNuovo(self, event):
 wxMessageBox("Hai premuto il tasto NUOVO")
```

I più attenti avranno certamente notato la doppia notazione *wx.Button/wxButton*. Gli autori della libreria stanno migrando le classi nel namespace *wx*, per ragioni legate alla compatibilità a ritroso è ancora possibile usare le classi con il prefisso *wx (wxFrame, wxButton, …)* del package *wxPython*. Per i nuovi progetti si suggerisce di dichiarare *import wx* al posto di *from wxPython.wx import **, ricordatevi però

LAYOUT

Con il termine layout ci si riferisce alla disposizione dei controlli in una finestra. wxPython presenta numerosi metodi per controllare tale disposizione: GridBagSizer, Anchors, Constraints, Layoutf, Sizers, risorse XML, etc.

di separare con un punto il prefisso *wx* ed il nome del metodo. *wxFrame* diventa *wx.Frame,* le altre classi vanno modificate di conseguenza, in caso contrario riceverete l'errore:

"NameError: name 'wx <NomeClasse>' is not defined".

Se avete bisogno di chiarirvi le idee potete dare un'occhiata agli script pubblicati sul CD.

UN ESEMPIO COMPLETO

Visto come collocare i controlli in una finestra e come rispondere agli eventi passiamo all'implementazione di un rudimentale web-browser. Il nostro obiettivo è la creazione di un'interfaccia portabile, perciò non opteremo per il componente ActiveX di Internet Explorer, non vogliamo vincolarci ai sistemi Windows. *wxPython* prevede una classe per la visualizzazione di pagine HTML: *wx.HtmlWindow.* Vi rimando alla documentazione di *wx.HtmlWindow* per quanto riguarda la personalizzazione dei formati grafici supportati, dell'interpretazione del codice HTML, etc. Nel file *esempio5.py* trovate il codice dello script *"ioProgrammo Browser"* (**Figura 7**). La resa di *wx.HtmlWindow* non è certo paragonabile a quella di browser quali Internet Explorer o Mozilla FireFox, per questo viene mostrato un messaggio in fase di avvio:

```
if wx.Platform == '__WXMSW__':
wx.MessageBox("Utilizzando wx.lib.iewin si ottiene una
  resa nettamente migliore!","Per gli utenti Windows:")
```

wx.Platform restituisce informazioni sulla piattaforma corrente (**Tabella 1**) ed è importante per caratterizzare l'esecuzione dei propri script su sistemi diversi.

Sistema	Valore di wx.Platform
Windows	__WXMSW__
GTK (Unix/Linux)	__WXGTK__
X11 (Unix/Linux)	__WXX11__
Macintosh (MAC)	__WXMAC__

Tabella 1: aaaa

Il nostro script inizia con le clausole di importazione:

```
import wx
# La riga seguente ci consentirà di utilizzare
                                    wx.HtmlWindow
import wx.html as html
```

Evito di riportare il codice che genera i pulsanti in basso perché non introduce nessuna novità rispetto al paragrafo precedente. Nella classe *Finestra* trovia-

SUL WEB

http://www.wxpython.org
Il progetto WxPython in tutto il suo splendore.

http://www.python.org
Il sito ufficiale del linguaggio Python.

http://www.python.it
Risorse e guide in italiano su Python.

http://smeschini.altervista.org
Website dell'autore.

http://wxglade.sourceforge.net
Uno strumento per lo sviluppo rapido di interfacce.

http://spe.pycs.net
SPE è un IDE multipiattaforma integrato con wxGlade.

Fig. 7: Il web browser in tutto il suo splendore!

SVILUPPO RAPIDO CON WXGLADE

CREARE IL FRAME

1 **Dopo aver lanciato lo script wxglade.py dobbiamo aggiungere un frame al nostro progetto cliccando sul pulsante "Add a frame". Confermiamo le opzioni di default.**

NUOVO SLOT

2 **Selezioniamo la voce sizer_1 dalla struttura del progetto, cliccando con il pulsante destro si visualizza il menu da cui è necessario scegliere "Add slot".**

UN ALTRO SIZER

3 **Per ottenere un'interfaccia più complessa inseriamo altri controlli di tipo sizer all'interno di quelli esistenti. La figura mostra il risultato finale.**

mo un'istanza di *wx.HtmlWindow* denominata *Visualizzatore*:

```
# Controllo per la visualizzazione di pagineWeb
self.Visualizzatore = html.HtmlWindow(self, -1)
htmlbox = wx.BoxSizer(wx.VERTICAL)
# Nella parte alta viene visualizzata la pagina
htmlbox.Add(self.Visualizzatore, 2, wx.EXPAND)
```

Alla pressione del pulsante etichettato *"Vai"* si esegue il gestore di evento *OnVai*, il caricamento della pagina è effettuato dal metodo *LoadPage*. Il parametro *url* può fare riferimento ad un file presente sul disco rigido o ad un indirizzo Web:

```
# Event handler collegato al pulsante VAI
  def OnVai(self, event):
      # Indirizzo specificato dall'utente
    url = self.indirizzo.GetValue()
    if url != "":
      self.Visualizzatore.LoadPage(url)
    else:
wx.MessageBox("Specifica un indirizzo o un file!")
```

Dopo aver creato un'istanza di *wx.FileDialog* chiamata *dlg* bisogna invocare il metodo *ShowModal* per mostrare la finestra che permette la selezione dei file (**Figura 8**):

```
# Event handler collegato al pulsante "Carica File"
  def OnCaricaFile(self, event):
    dlg = wx.FileDialog(self, wildcard = '*.htm*', style=
                                                  wx.OPEN)
    if dlg.ShowModal():
      # fileHTML contiene il percorso del file scelto
      fileHTML = dlg.GetPath()
      self.Visualizzatore.LoadPage(fileHTML)
    dlg.Destroy()
```

La creazione manuale dell'interfaccia utente viene considerata, a ragione, un'attività noiosa. Esistono, per nostra fortuna, molti strumenti che agevolano la vita dello sviluppatore *wxPython: wxDesigner, wxGlade, XRCed, Boa Constructor*, etc. Nel riquadro **Sviluppo rapido** sono descritti i passaggi che garantiscono, grazie a *wxGlade*, un approccio indolore a *wxPython*.

CONCLUSIONI

Un'importante fonte di documentazione per *wx Python* è rappresentata dagli script dimostrativi inclusi in uno dei due pacchetti della distribuzione ufficiale. La sintassi del linguaggio Python e la sua libreria standard ricca di funzioni contribuiscono a renderlo uno strumento ideale per l'uso in contesti eterogenei, non meraviglia dunque il fatto che la sua popolarità sia in forte crescita. Sui prossimi numeri di ioProgrammo troveranno spazio altri approfondimenti su Python, nel frattempo potreste dedicarvi alla lettura di una delle tante guide introduttive.
A presto!

Salvatore Meschini

Fig. 8: ***Il dialog per la selezione dei file varia in base al sistema operativo***

AGGIUNGERE CONTROLLI

4 **Abbiamo creato la struttura dell'interfaccia grafica, ora disponiamo i controlli selezionandoli tra quelli disponibili e scegliendo lo slot di destinazione.**

LE PROPRIETÀ

5 **Ogni controllo (pulsante, griglia, ...) ha delle proprietà che ne modificano l'aspetto ed il comportamento. Le proprietà sono suddivise in tre linguette distinte.**

GENERARE IL CODICE

6 **È tutto pronto! Finalmente possiamo generare il codice Python basato sulla libreria wxPython: scegliamo un percorso e premiamo il pulsante "Generate code".**

Tecnologia XML e calcolo combinatorio

Creare il calendario di un campionato di calcio

Gestire un problema di calcolo combinatorio, come la creazione del calendario di un campionato di calcio, con l'utilizzo della tecnologia XML, di alcuni oggetti ADODB e di un database Access

Discussione aperta all'indirizzo
http://forum.ioprogrammo.net/thread.php?threadid=4047&boardid=13

Utilizza questo spazio per le tue annotazioni

Nel forum di ioProgrammo all'indirizzo *http://forum.ioprogrammo.net/thread.php?threadid=4047&boardid=13* è nata di recente un'interessante discussione su come creare un calendario di incontri per un campionato di calcio. Il problema, apparentemente semplice è apparso invece decisamente complesso e interessante tanto che ha scatenato una serie di proposte mirate a produrre un algoritmo ottimizzato. In realtà si tratta di applicare un problema di calcolo combinatorio alle strutture esistenti in programmazione. Inoltre trovando una soluzione generale si può applicare lo stesso algoritmo per ogni problema che preveda la generazione di coppie di combinazioni. Ad esempio la generazione di un orario scolastico. Perciò abbiamo deciso di proporre una soluzione completa di implementazione in Visual Basic. Le ottimizzazioni possibili sono comunque infinite, tanto che la discussione rimane aperta ad ogni nuovo contributo. In questa miniserie di due articoli proporremo una soluzione che da un lato implementa l'algoritmo di calcolo combinatorio, dall'altro sceglie XML come struttura di supporto al contenuto informativo. In questo primo numero parleremo quasi esclusivamente di XML mentre nel secondo parleremo dell'algoritmo di calcolo combinatorio. I due articoli sono comunque autoconclusivi perciò potrete leggerli comunque in maniera indipendente. In questo primo articolo avrete occasione di imparare come Visual Basic gestisce i file XML.

Fig. 1: Il form principale dell'applicazione

L'ALGORITMO DI CALCOLO

Considerato un certo numero di squadre, bisogna trovare tutti gli accoppiamenti (coppie di squadre) in modo che siano soddisfatti determinati vincoli. Dalla teoria del calcolo combinatorio si deduce che il numero totale di coppie, e quindi di partite, per questo problema si può ricavare con le combinazioni semplici di n elementi distinti presi a gruppi di due. Nel caso di n squadre, bisogna considerare i seguenti punti.

1. I turni di campionato corrisponderanno a *n-1;*
2. Per ogni turno saranno disputate *n/2;*
3. Inoltre è necessario considerare i seguenti vincoli:
   - una coppia di squadre può incontrarsi se non l'ha gia fatto nei turni precedenti;
   - se una squadra nel turno corrente gioca in casa nel successivo deve giocare fuori casa;
   - altri vincoli che dipendono dalle necessità delle singolo squadre …
4. Inoltre se n è dispari bisogna prevedere un turno di riposo per ogni squadra e di conseguenza le partite per turno saranno (n-1)/2.

S'intuisce che per implementare l'algoritmo bisogna utilizzare una struttura dati complessa che permette di tracciare le scelte fatte. A tal fine si è preferito utilizzare un albero XML dato che la sua struttura si presta a questo tipo di approccio e anche perché dopo che il calendario è creato può essere facilmen-

te modificato ed esportato su database, Web ecc.

IL DATABASE DI SUPPORTO

In questo paragrafo è descritto il database, che contiene i dati delle squadre, nominato calcio .mdb. Esso contiene una sola tabella nominata squadra la cui struttura è presentata nella **Tabella 1**.

Squadra

Nome Campo	Tipo Dato
codiceQ	Intero
nomeQ	Testo
Calendario	Memo
Note	Memo

Inizialmente squadra contiene soltanto i dati delle squadre. L'applicazione riceverà i dati in input dal database MDS e produrrà in output un calendario in formato XML

L'ALBERO XML DELLE SQUADRE

In questo paragrafo si descrive la struttura del documento XML alla base dell'applicazione. I dati del campionato di calcio si possono strutturale utilizzando i tag: *<campionato>*, *<squadre>*, *<squadra>*, *<codiceSQ>*, *<nomeSQ>* e *<partite>*.

Campionato è il tag *radice*, *squadre* ha come nodi figli *squadra* che a sua volta ha come figli il *codice* della squadra *(codiceSQ)*, il *nome* della squadra *(nomeSQ)* e le *partite* da disputare (descritti con i tag *giornata*, *SQavversaria*). Con la sintassi XML l'albero si descrive nel seguente modo.

```
<campionato>
 <squadre>
  <squadra>
    <codiceSQ>  1  </codiceSQ>
    <nomeSQ>  Roma </nomeSQ>
    <partite>
      <giornata> 1 </giornata >
```

ELEMENTO NODO

L'elemento più comune in un Documento XML e nel DOM è il nodo cioè *IXMLDOMNode*. Esso supporta varie caratteristiche tra le quali si evidenziano:

- ***Text*, contiene il testo inserito nel nodo e nel sotto albero associato;**
- ***NodeName*, restituisce il nome del nodo;**
- ***FirstChild*, contiene il primo nodo figlio del nodo corrente (quindi restituisce un elemento nodo);**
- ***SelectNodes*, in base al percorso XPath, specificato, restituisce la lista dei nodi che vi corrispondono; quindi restituisce una collezione di nodi, manipolabile con gli strumenti classici di Visual Basic.**
- ***SelectSingleNode*, sempre in base ad una query XPath permette di selezionare un singolo nodo. La sintassi per utilizzare la *SelectNodes* è la seguente:**

```
Set objXMLDOMNodeList =
oXMLDOMNode.selectNodes
(query XPath)
```

COME CALCOLARE LE COMBINAZIONI SEMPLICI DI N ELEMENTI PRESI A K A K

Il numero di partite disputate in un campionato di calcio, di n squadre, è pari al numero di combinazioni semplici di n elementi distinti presi a gruppi di k (con k=2). La formula per calcolare le combinazioni semplici è la seguente:

$$Cn,k = \frac{n\cdot(n-1)\cdot(n-2)\cdot\ldots\cdot(n-k+1)}{k!}$$

Dove k! è il fattoriale di k, cioè $k! = 1\cdot2\cdot3\cdot\ldots\cdot(k-1)\cdot k$. Nel caso in cui K=2 (gruppi di due squadre) la formula precedente diventa n(n-1)/2. Allora si deduce che in un campionato con n squadre si disputano al massimo n(n-1)/2 partite (nel girone d'andata), che in un turno (giornata) si possono disputare al massimo n/2 partite e che per disputare le n(n-1)/2 partite occorrono n-1 turni. Per esempio se n=12 il numero totale di partite è 12*11/2=66, le partire per turno sono 6 e i turni sono 66/6=11.
Per implementare la formula delle combinazioni semplici si può procedere nel seguente modo:

1 **Creare un nuovo progetto e sul form1 inserire un pulsante e due textbox.**

```
Private Sub Calcola_Click()
  Dim val As Long
  val = CStr(Prod(Text1, Text2) / Fatt(Text2))
  MsgBox "Le combinazione di " + CStr(Text1)
   + " elementi presi a " + CStr(Text2) _ & " alla volta
                                    sono: " + CStr(val)
  End Sub
```

2 **Nella Click del CommandButton (*nominato Calcola*) inserire il codice. Con un MsgBox vengono invocate le procedure Fatt e Prod. La Fatt valuta il fattoriale di un numero, la Prod valuta il numeratore della formula delle combinazioni.**

```
Public Function Fatt(n As Long)
  Dim i As Integer
  Dim f As Long
  f = 1
  For i = 2 To n
    f = f * i
  Next
  Fatt = f
End Function
Public Function Prod(n As Long, k As Long)
  Dim i As Integer
  Dim p As Long
  p = 1
  For i = n - k + 1 To n
    p = p * i
  Next
  Prod = p
End Function
```

3 **Il codice da predisporre per le procedure Fatt e Prod.**

```
    < SQavversaria > Juventus </SQavversaria >
...
   </partite>
  </squadra>
...
</squadre>
</campionato>
```

Quello scritto sopra deve essere ripetuto per ogni squadra e per ogni giornata. Notare che per ogni tag (detto *start tag*) è presente il tag di chiusura (detto *end tag*) e che non ci sono sovrapposizioni tra i tag. Attenzione, inoltre, alle maiuscole perché i tag sono case sensitive per questo <*Campionato*> è diverso da <*campionato*>.

Fig. 2: ***Il documento XML del campionato***

XSL

L'XSL è una famiglia di raccomandazioni per la definizione di trasformazioni e presentazioni XML. Essa è composta da tre parti: XSTL (XSL Transformations) che è un linguaggio per trasformazioni XML; XPath (XML Path Language) che sono delle espressioni usate per interrogare i documenti XML (con una sintassi simile a quella utilizzata per navigare le directory del file system) e XSL-FO (XSL Formatting Object) un vocabolario XML per specificare una semantica di formattazione per documenti XML.

QUERY XPATH

La notazione *XPath (XML Path Language)* è usata per interrogare un documento XML, essa è supportato dal Parser *MSXML* dentro le XSLT e con gli elementi *selectNodes* e *selectSingleNode*. L'*XPath* è una notazione dichiarativa piuttosto che procedurale per questo con essa si descrivono soprattutto delle relazioni gerarchiche tra nodi. Per esempio con il percorso *"calendario/squadre/ squadra"* si indica che si vogliono ricercare i nodi squadra contenuti in squadre. In una query XPath possono essere utilizzati vari operatori e caratteri speciali, alcuni di questi sono specificati nella **Tabella 2.**

Operatore	Descrizione
/	*Operatore figlio*, permette di selezionare i figli di un nodo
//	*Operatore Recursive descent*, quando è specificato all'inizio della query indica che si deve discendere l'albero a partire dalla radice
.	Indica il contesto corrente
@	Operatore per indicare o selezionare gli attributi dei nodi
*	Carattere jolly da utilizzare per selezionare tutti gli elementi.
[ ]	Operatore utilizzato per selezionare un elemento specifico di una collezione.

Tabella 2

Per esempio per selezionare tutti i nodi squadra si possono utilizzare le seguenti righe di codice.

```
Dim comlist As IXMLDOMNodeList
Dim num As Integer
```

PARSER MSXML E MODELLO DOM

In questo paragrafo vengono fatte considerazioni sui seguenti argomenti:

1. **Documenti e alberi XML;**
2. **Parser MSXML;**
3. **DOM (Document Object Model), il modello ad oggetti per manipolare un documento XML;**
4. **Gli oggetti del DOM utilizzati negli esempi.**

Un documento XML (file testo con estensione XML), nei sistemi operativi Windows, è interpretato dal Parser MSXML (Microsoft XML Parser). MSXML.DLL è una libreria presente in Internet Explorer già dalla versione 4. Il Parser, in parole povere, è un software che legge un documento XML lo interpreta e permette l'accesso al suo contenuto, esso, inoltre, controlla gli errori sintattici e li mostra, nel caso ne rilevi qualcuno. Un documento XML può essere elaborato secondo due approcci: "Event-based" e "Navigazionale". Il primo approccio si basa su SAX (Simple API for XML): API che permette di scrivere applicazioni che possono leggere dati da un documento XML. Il secondo si basa su Document Object Model (DOM): un modello che permette a programmi e script di leggere e modificare il contenuto, la struttura e lo stile di un documento XML. In particolare il DOM fornisce un set standard di oggetti per presentare documenti HTML e XML e per combinare elementi dei due mondi; quindi facilita la pubblicazione, su pagine Web, dei dati contenuti in un file XML. Attraverso il DOM in memoria è costruita una rappresentazione ad albero del documento, tramite la quale si può accedere in modo random al suo contenuto.
I principali elementi del DOM utilizzati nell'articolo sono: *DOMDocument, IXMLDOMElement, IXMLDOMNode, IXMLDOMNodeList* e *getElementsByTagName.*
***DOMDocument* è il livello superiore di un albero XML e contiene gli strumenti per creare e ritrovare gli elementi sottostanti. *IXMLDOMElement* rappresenta un elemento XML generalizzato, esso tra l'altro, fornisce i metodi che permettono di gestire gli attributi. *IXMLDOMNodeList,* invece, è una collezione di nodi con una proprietà e tre metodi. La proprietà è la *length,* che indica il numero di *items* (elementi) presenti nella collezione; i metodi, invece sono: *item,* che permette di accedere ai singoli nodi attraverso degli indici; *NextNode,* che restituisce il nodo successivo al corrente; *reset,* che resetta il contatore di posizione e lo imposta alla posizione precedente al primo nodo, così che una chiamata a *NextNode* restituise il primo item della lista. Naturalmente per percorrere i nodi della collezione è possibile utilizzare l'istruzione *For... each. GetElementsByTagName* restituisce una collezione di elementi che hanno come nome il tag specificato come parametro.**

```
Set comlist = xmldocument.documentElement
                    .selectNodes("//squadre/*")
num = comlist.length
```

Num contiene il numero di squadre inserite nell'albero.

COSTRUIRE L'ALBERO DEL CAMPIONATO

Nell'applicazione per creare l'albero, completo, del campionato si procede nel modo seguente.

1. Si crea l'albero XML predisponendo i nodi *<campionato>*, *<squadre>*, ecc.
2. Per ogni giornata, attraverso l'algoritmo che descriveremo nel successivo appuntamento, si valutano gli avversari (*SQavversari*) delle squadre.
3. Si creano i rami *<partite>* con i dati degli *SQavversari.*

Di seguito è presentato il codice che realizza il primo passo e l'archiviazione. Questo va inserito nei pulsante *CreaAlberoXml* e *Salva* del form principale introdotto nell'esempio precedente.

```
Private Sub creaalberoxml_Click()
 On Error GoTo errori
 Dim Rst As Recordset
 Const radice = "campionato"
 Dim docroot As IXMLDOMElement
 Dim figlio As IXMLDOMElement
 Dim Qpadre As IXMLDOMElement
 Set xmldocument = New DOMDocument
 xmldocument.async = False
 'non download asincrono
 Set Rst = eseguiquery("select * from squadra")
 Set docroot = xmldocument.createElement(radice)
 xmldocument.appendChild docroot
 Set figlio = xmldocument.createNode(NODE_ELEMENT,
                                          "squadre", "")
 xmldocument.selectSingleNode(radice).appendChild figlio
```

NOTA

Per caricare un documento XML in un oggetto *DOMDocument* si può usare il metodo *Load*

```
boolValue = XMLDocument
                .load(xmlSource)
```

***Xmlsource* è una stringa che contiene l'URL della posizione del file. *BoolValue* è *true* se il file esiste, *null* se non esiste.**

CREARE IL CALENDARIO - PRIMI PASSI

1 Creare un nuovo progetto che referenzi le seguenti librerie: *MSHFlxGd.ocx* (Microsoft Hierarchical FlexGrid), *shdocvw.dll* (Microsoft Internet Controls), *MSXML4.dll* (Microsoft XML, v4.0) oppure la *MSXML3.dll*

2 Disporre sul *form1* un controllo MSHFlexGrid, un WebBrowser e 5 CommandButton, nominati: *Caricasquadre, CreAlberoXml, Genera, Salvaalbero, Interroga*, che utilizzeremo per dare controllo all'utente

```
Dim n As Integer
 'numero di squadre partecipanti
Dim RstSquadre As ADODB.Recordset
```

3 Dichiarare le seguenti variabili globali. Ci serviranno rispettivamente per contenere il numero delle squadre e il recordset contenente i dati occorrenti per gestire correttamente il nostro calendario

```
Private Sub CaricaSQ_Click()
 MSHFlexGrid1.FixedCols = 0
 Set RstSquadre = eseguiquery("select
            codiceSQ, nomeSQ from squadra")
 If RstSquadre.RecordCount <= 0 Then
  MsgBox "Database vuoto", vbCritical,
                              "Attenzione"
  Exit Sub
 Else
  MSHFlexGrid1.AllowUserResizing =
                              flexResizeBoth
  MSHFlexGrid1.Appearance = flex3D
  Set MSHFlexGrid1.DataSource =
                      RstSquadre.DataSource
  n = RstSquadre.RecordCount
 End If
End Sub
Private Function eseguiquery(query As
                        String) As Recordset
 Dim strcnn As String
 Dim Rst As ADODB.Recordset
 strcnn = "Provider=microsoft.jet.oledb.4.0;"
 _
 & "Data Source=" + App.Path + "\calcio.mdb;
 "
 Set Rst = New ADODB.Recordset
 Rst.CursorType = adOpenKeyset
 Rst.LockType = adLockOptimistic
 Rst.Open query, strcnn, , , adCmdText
 Set eseguiquery = Rst
```

4 Predisporre la procedura seguente nel pulsante *CaricaSquadre*. Si noti che *n* rappresenta il numero di squadre presenti nel database e che la *CaricaSQ* invoca la procedura *eseguiquery*. Quest'ultima riceve come parametro un comando SQL, che invia al database, e restituisce un recordset popolato con i record recuperati dal database. La *eseguiquery* è utilizzata anche dalla procedura che imposta, nel campo calendario, i dati dell'albero XML

5 Nella figura è visualizzato lo stato del Form dopo aver premuto il pulsante *CaricaSquadre*. Le squadre compaiono sulla sinistra nell'apposito controllo che abbiamo definito in precedenza

SAVE E LOAD

Per salvare un documento XML, creato con un oggetto DOMDocument, si deve usare il metodo save che presenta la seguente sintassi:

```
XMLDocument.save
              (Destinazione)
```

***Destinazione* può rappresentare il nome di un file, un oggetto *Response (ASP)*, un *DOMDocument* oppure un oggetto *COM* con particolari caratteristiche. Per questo metodo nella guida all'MSXML è presente una tabella che illustra tutti i possibili valori.**

```
Set figlio = Nothing
Do While Not Rst.EOF
 Set Qpadre = xmldocument.createNode(
                     NODE_ELEMENT, "squadra", "")
 xmldocument.selectSingleNode(radice + "/squadre"
                                  ).appendChild Qpadre
 For i = 0 To 1
  Set figlio = xmldocument.createNode("element",
                                 Rst.Fields(i).Name, "")
  figlio.Text = Rst.Fields(i).Value
  Qpadre.appendChild figlio
  Set figlio = Nothing
 Next
 Set figlio = xmldocument.createNode("element",
                                          "partite", "")
 Qpadre.appendChild figlio
 Rst.MoveNext
Loop
 xmldocument.Save (App.Path + "\" + "solosquadre.xml")
 WebBrowser1.Navigate App.Path + "\" + "solosquadre.xml"
 Rst.Close
Exit Sub
errori:
 MsgBox Err.Description
 Err.Clear
End Sub
```

La *creaalberoxml* prima interroga il database, utilizzando la *eseguiquery*, e poi crea l'albero a partire dalla radice (oggetto *docroot*). Alla radice viene "appeso" (con *appendChild*) il nodo squadre e i nodi squadra creati con i dati contenuti nel recordset. Notare che *Rst. Fields (i).name* è il nome del campo della tabella squadre e che *Rst .Fields(i).value* è il valore del campo. Nella procedura precedente è anche previsto il codice che permette di salvare l'albero XML (delle sole squadre, senza partite) in un file XML esterno (nominato *solosquadre.xml*). Questo, infine, attraverso il metodo *Navigate* è caricato nel WebBrowser. Nel pulsante *salvaalbero*, invece, bisogna inserire il seguente codice.

Fig. 3: **Il form principale dopo aver caricato le squadre e l'albero**

Fig. 4: **Il form per interrogare i documenti XML e il database bero**

```
Private Sub Salva_Click()
 Dim com As IXMLDOMElement
 Dim codiceSQ As IXMLDOMNode
 Dim squadra As IXMLDOMElement
 Dim i As Integer
 Dim objNodeListQ As IXMLDOMNodeList
 Set objNodeListQ = xmldocument
                    .getElementsByTagName("squadra")
 For i = 0 To (objNodeListQ.length - 1)
   eseguiquery ("update squadra set calendario = '" +
     objNodeListQ.Item(i).xml + "' where " & " codiceSQ =
          " + objNodeListQ.Item(i).childNodes(0).Text)
 Next
  xmldocument.Save (App.Path + "\" + "calendario.xml")
End Sub
```

La *Salva* consente di salvare il ramo <*squadra*> nel campo calendario della tabella squadra. In particolare con questo primo esempio è salvata la seguente stringa:

```
<squadra>
 <codiceSQ>1</codiceSQ>
 <nomeSQ>Inter</nomeSQ>
 <partite/>
</squadra>
```

Il tag <*partite*> verrà popolato nel successivo appuntamento!

WEB BROWSER

L'oggetto WebBrowser permette di utilizzare alcune caratteristiche di Internet Explorer, in particolare permette di consultare documenti Web e tenerne traccia (nella cronologia). Rispetto a Internet Explorer, però, non mostra i vari tool (Address Bar, tool dei bottoni ecc.) necessari per navigare sui siti web, questi devono essere previsti e gestiti dal programmatore. Un file XML può essere caricato nel WebBrowser utilizzando il metodo Navigate che permette di caricare il file specificato attraverso il suo URL (o path locale). Per inserire in un progetto un oggetto WebBrowser bisogna includere la libreria *shdocvw.dll (Microsoft Internet Controls).*

CONCLUSIONI

L'applicazione *"campionato di calcio"* verrà completata nel successivo appuntamento, dove si descriverà l'algoritmo di calcolo che permette di valutare le partite per ogni turno, l'eventuali turno di riposo, le partite da disputare in casa e fuori casa ecc. Nel successivo appuntamento, inoltre, non mancheranno i richiami sulla tecnologia XML, sull'HTML e sulle applicazioni Office, seguiteci!

Massimo Autiero

Architetture distribuite con Java

Messaggi in una bottiglia

Realizzeremo un sistema di rilevazione delle presenze aziendali basato sul controllo di un badge. I nostri soli strumenti saranno Java e Mantaray

Nel numero di ioProgrammo di gennaio 2005 è stato introdotto Mantaray, un nuovo strumento di messaggistica aziendale conforme alle specifiche JMS, l'API standard della piattaforma Java per le architetture orientate ai messaggi. Nel precedente articolo si è parlato di JMS, Mantaray e interazione con PHP; in questo numero vedremo un'applicazione pratica di questa tecnologia con un esempio in Java, un semplice sistema di controllo presenze distribuito (nelle **Figure 1** e **2** è illustrata una tipica sessione di comunicazione tra client e server).

```
D:\WINNT\system32\cmd.exe
F:\projct\c6>java -DmantaHome=./client -cp lib/antlr.jar;lib/bsh-1.2b3.jar;lib/c
link-1.5.0.jar;lib/commons-logging.jar;lib/concurrent.jar;lib/inria.jar;lib/jms.
jar;lib/junit.jar;lib/log4j-1.2.8.jar;lib/manta.jar;lib/mx4j-jmx.jar;lib/mx4j-re
mote.jar;lib/mx4j-tools.jar;lib/mx4j.jar;lib/mysql-connector-java-3.0.11-stable-
bin.jar;lib/xerces.jar;classes/ it.bigatti.c6.BadgeReader
property 'mantaHome'=./client
property 'configurationFolder'=./client/config/
./client/config/ found.
property 'defaultConfigurationFile'=./client/config//default_config.params
./client/config//default_config.params found.
property 'agentConfigurationFile'=./client/config//component_config.params
./client/config//component_config.params found.
property 'worldXmlFilePath'=./client/config//world.xml
./client/config//world.xml found.
property 'pesistentFolder'=./client/persistent/
./client/persistent/ found.
Pronti
[input]> 8787857438
verificaBadge() 8787857438
[input]> [send] BADGE: 8787857438
[receive] BADGE: BADGE: 8787857438 KO
000000000050001
[input]> verificaBadge() 000000000050001
[send] BADGE: 000000000050001
[receive] BADGE: BADGE: 000000000050001 OK
[input]> verificaBadge() null
```

Fig. 1: ***Il client di C6 in esecuzione***

```
D:\WINNT\system32\cmd.exe
./server/config//component_config.params found.
property 'worldXmlFilePath'=./server/config//world.xml
./server/config//world.xml found.
property 'pesistentFolder'=./server/persistent/
./server/persistent/ found.
server...
server...
[receive]BADGE: 8787857438
SELECT id FROM badges WHERE  codice_badge='8787857438' AND data_scadenza>=NOW()
AND valida=1
[send] BADGE: BADGE: 8787857438 KO
server...
server...
[receive]BADGE: 000000000050001
SELECT id FROM badges WHERE  codice_badge='000000000050001' AND data_scadenza>=N
OW() AND valida=1
[send] BADGE: BADGE: 000000000050001 OK
INSERT INTO registrazioni VALUES ('000000000050001',NOW(),'E')
server...
[receive]BADGE: null
SELECT id FROM badges WHERE  codice_badge='null' AND data_scadenza>=NOW() AND va
lida=1
```

Fig. 2: ***Il server di C6 in esecuzione***

PRESENTI ALL'APPELLO

Il programma di controllo delle presenze sviluppato nel corso di questo articolo, chiamato C6 (ci sei?) serve per supervisionare gli ingressi ad un ambiente controllato, tramite l'uso di badge (tesserini) personali. Ciascun badge ha un codice univoco ed è assegnato ad una persona. Quando il badge viene passato nel lettore, il sistema controlla che sia valido; se lo è, il cancelletto o la porta di accesso vengono sbloccati per consentire alla persona di entrare. Contestualmente, il sistema registra l'avvenuto ingresso nel sistema da parte di quel badge, insieme alla data ed all'ora. Ovviamente lo stesso meccanismo si applica in uscita, con la memorizzazione della data/ora di uscita dall'ambiente protetto. Il programma C6 si compone di tre elementi fondamentali: un programma di acquisizione di un codice badge, che simula un dispositivo di lettura del badge personale, questo programma invia il numero di badge al server, un programma server in ascolto dei messaggi del programma di acquisizione badge. Alla loro ricezione restituisce un messaggio al mittente indicando se la tessera è valida, non scaduta e non bloccata. Nello stesso momento, viene registrato l'ingresso di quel badge nell'ambiente supervisionato dal programma. Per gestire queste informazioni, il programma server interagisce con un database MySQL dove è presente l'anagrafica dei badge e l'archivio delle entrate e delle uscite. La struttura della tabella BADGES è la seguente:

```
CREATE TABLE badges (id int(11) NOT NULL
   auto_increment, codice_badge varchar(15) NOT NULL
        default '', nome varchar(30) NOT NULL default '',
                cognome varchar(30) NOT NULL default '',
```

id.	identificativo univoco di riga;
codice_badge.	Codice del badge, anch'esso univoco;
nome.	Nome della persona a cui è associato il badge;
cognome.	Cognome della persona a cui è associato il badge;
data_registrazione.	Data in cui l'utente è stato registrato nel sistema;
data_scadenza.	Data di scadenza della registrazione;
valida.	Se vale zero, il badge è stato bloccato per qualche motivo.

Tabella 1

NOTA

MANTARAY E MAC OS X: GIOIE E DOLORI

La libreria di messaging è stata provata su Mac OS X e sono stati riscontrati alcuni piccoli problemi: il programma di Chat incluso negli esempi sembrava perdere dei messaggi ogni tanto. Con Monitoraggio Attività sembrava anche di rilevare un impegno del 20% del processore a processo. Ma forse questi sono problemi di gioventù che saranno risolti in fase di testing del software.

```
data_registrazione datetime NOT NULL default '0000-
00-00 00:00:00', data_scadenza datetime NOT NULL
default '0000-00-00 00:00:00', valida int(11) NOT
NULL default '0', PRIMARY KEY (id));
```

i campi sono elencati in **Tabella 1.**
Ogni entrata ed uscita gestita dal sistema viene memorizzata come record nella tabella registrazioni, che è strutturata come segue:

```
CREATE TABLE registrazioni (codice_badge varchar(15)
NOT NULL default '', data_registrazione datetime NOT
NULL default '0000-00-00 00:00:00',
entrata_uscita char(1) NOT NULL default '');
```

per ogni evento viene memorizzato il codice del badge e la data ed ora correnti. Il flag *entrata_uscita* vale *E* se l'utente sta entrando oppure *U* se sta uscendo.

SVILUPPARE IL CLIENT

A questo punto è possibile iniziare lo sviluppo del programma client, che è composto dalla sola classe *BadgeReader.* Nel costruttore di questa è presente tutto il codice di inizializzazione del sistema di messaggistica. Inoltre è presente la logica del programma. Le operazioni svolte includono la creazione di una factory di connessioni, che permette di ottenere un oggetto iniziale per interagire con le code:

```
QueueConnectionFactory conFactory =
        (QueueConnectionFactory) new
        MantaQueueConnectionFactory();
//Crea una connessione alla coda
con = conFactory.createQueueConnection();
```

con la connessione si ottengono degli oggetti sessione, che identificano un insieme di operazioni con le code. Questi oggetti sono richiesti dalle API JMS:

```
//Ottiene le sessioni di invio e ricezione
sendSession = (QueueSession)
con.createQueueSession(false,
        Session.AUTO_ACKNOWLEDGE);
receiveSession = (QueueSession)
con.createQueueSession(false,
        Session.AUTO_ACKNOWLEDGE);
```

le code di invio e ricezione vengono create specificandone il nome. Nella classe BadgeReader la classe C6Constants contiene i nomi delle code:

```
//Crea la coda di invio
javax.jms.Queue sendQueue = sendSession.createQueue(
        C6Constants.SEND_QUEUE_NAME);
sender = sendSession.createSender(sendQueue);
//Crea la coda di ricezione
javax.jms.Queue receiveQueue = receiveSession.createQueue(
        C6Constants.RECEIVE_QUEUE_NAME);
javax.jms.QueueReceiver qReceiver =
        receiveSession.createReceiver(receiveQueue);
```

L'interfaccia C6Constants è la seguente:

```
package it.bigatti.c6;
public interface C6Constants {
  final String SEND_QUEUE_NAME = "BadgeOut";
  final String RECEIVE_QUEUE_NAME = "BadgeIn";
}
```

raccogliendo queste costanti in un unico punto si rende più semplice la modifica dei nomi delle code, in quanto sono centralizzati in un unico punto.
Tornando al costruttore di BadgeReader, una volta create le code è possibile installare un ascoltatore di eventi sulla coda in arrivo, in modo da elaborare le risposte del server. Il server di C6 non fa altro che ritornare una stringa contenente il codice del badge per cui si è fatta la richiesta ed un OK/KO ad indicare la validità o meno dello stesso:

```
//Aggiunge l'ascoltatore di eventi per i messaggi in arrivo
qReceiver.setMessageListener( new MessageListener() {
  public void onMessage(Message msg) {
    TextMessage tmsg = (TextMessage)msg;
```

COS'È MANTARAY?

Mantaray è un sistema di messaggistica open source, dotato di una architettura completamente distribuita. Diversamente da altre soluzioni, Mantaray non richiede un server centrale (o broker), elemento che gli dona due caratteristiche fondamentali. La prima è l'assoluta mancanza di punti di congestione, utile a gestire un carico di messaggi elevato tipico delle grandi architetture software che ne fanno uso. La seconda è la mancanza di punti di debolezza: le architetture basate su server centrale possono cadere se questo non funziona correttamente. In un sistema distribuito come Mantaray, utilizzando una progettazione accorta, è possibile evitare che la caduta di un singolo elemento blocchi l'intero sistema. Proprio come una nave Borg!
Mantaray dispone di interfacce di programmazione conformi allo standard JMS, dunque il software è facilmente interfacciabile dai programmatori Java che conoscono J2EE. In questa tecnologia sono presenti due tipologie di canali di comunicazione: le code ed i topic. Nel primo caso, un'applicazione invia messaggi su una coda, mentre un altro processo si occupa di riceverli. Chi invia un messaggio è detto produttore, chi lo riceve, consumatore. Ci possono essere un numero illimitato di produttori e consumatori su una singola coda, come un numero illimitato di code e di messaggi all'interno di esse. Uno specifico messaggio è inviato esplicitamente ad un consumatore ed una volta che questi lo ha ricevuto il messaggio viene eliminato.
L'altra modalità presente in JMS sono i topic (argomenti). Anche in questo caso si hanno produttori e consumatori, ma con la differenza che tutti i consumatori ricevono tutti i messaggi inviati ad un topic. Anche se un consumatore è offline, questo riceverà il messaggio una volta attivo. Attendo tue indicazioni. ovviamente puoi togliere il riferimento a Star Trek se lo ritieni inappropriato. ora termino la verifica delle bozze e ti faccio sapere

```
    String line;
    try {
        line = tmsg.getText();
        System.out.println( "[receive] " + line );
    } catch (JMSException e) {
        e.printStackTrace(); }
  }
});
```

questo pezzo di codice non fa altro che stampare su console il messaggio ricevuto dal server. Ogni qual volta viene ricevuto un messaggio, infatti, viene invocato il metodo onMessage(). A questo viene passato un parametro di tipo *Message*, che viene convertito in *TextMessage*. Questa operazione è possibile in quanto in fase di progettazione è stato deciso che C6 comunica solo con messaggi testuali. Dall'oggetto *TextMessage* viene estratto il testo del messaggio, attraverso il metodo *getText()*.

INVIARE LE RICHIESTE

Una volta inizializzato il sistema delle code viene lanciato il metodo *leggiEScrivi()* che implementa un ciclo infinito. All'interno di questo viene letto da tastiera il codice del badge. Per ogni codice viene invocato il metodo *verficaBadge()*:

```
void leggiEScrivi() throws JMSException, IOException {
  BufferedReader stdin = new BufferedReader(
                 new InputStreamReader( System.in ) );
  con.start();
  System.out.println("Pronti");
  while( true ) {
    System.out.print("[input]> ");
    String line = stdin.readLine();
    verificaBadge( line ); }
}
```

il metodo *verificaBadge()* non fa altro che creare un

COME FUNZIONA

Il sistema si basa su Mantaray, che come già detto è un architettura basata su messaggi. Mantaray deve essere installato sul lettore di badge e sul server che analizza il controllo delle presenza. Sul lettore di badge mantaray verrà configurato con due code "BadgeIn" e "BadgeOut". L'introduzione del tesserino nel lettore provocherà l'invio di un messaggio nella coda BadgeOut che provvederà a spedirlo al server. Nella coda BadgeIn verranno conservati invece i messaggi ricevuti dal server. Il server avrà anche esso due code "BadgeIn" e "BadgeOut" che rispettivamente riceveranno i messaggi dal lettore e spediranno messaggi di risposta al lettore. In sintesi l'applicazione potrebbe essere immaginata con un dialogo fra client e server.

- **l'utente inserisce un tesserino nel badge, viene spedito un messaggio al server tramite la coda BadgeOut**
- **Il server riceve il messaggio, controlla che il tesserino sia valido e spedisce un messaggio di Ok come risposta al lettore**
- **Il lettore riceve il messaggio, stampa a video l'autorizzazione e provvede ad aprire il cancello d'entrata.**

INSTALLAZIONE DI MANTARAY

L'installazione del software è tutto sommato semplice, anche se nasconde qualche insidia, specie se si desidera verificare il funzionamento della libreria sulla stessa macchina, e non su computer differenti. La procedura di installazione di Mantaray sulla stessa macchina prevede i seguenti passi:

1 OTTENIMENTO DEL SOFTWARE - Scaricare il file che contiene il pacchetto compilato dal CD di ioProgrammo, oppure dal sito Web del progetto (si veda il box "Casa dolce Casa"). Al momento della scrittura di questo articolo, l'ultima versione disponibile è la 1.4.1, contenuta nel file *mantaray_ 1.4.1_bin.zip.*

2 SCOMPATTAMENTO ED INSTALLAZIONE - Decomprimere il file in una directory del disco rigido, ad esempio ~/tools/mantaray1. Si noti che la directory è numerata, questo servirà al passaggio successivo.

3 DUPLICAZIONE DELLA CARTELLA - Copiare la directory dove è stato installato Mantaray in un'altra, ad esempio *~/tools/mantaray2.* In questo momento sono presenti due directory identiche: *mantaray1* e *mantaray2.* Questo passo è essenziale per il funzionamento del software sulla stessa macchina: ciascuna istanza di software deve essere infatti configurata in modo diverso.

4 CONFIGURAZIONE SECONDA INSTALLAZIONE - A questo punto è possibile configurare la seconda istanza del software, cambiando due file. Il primo è *component_config .params:*

```
net.this.agent.name=myAgentNameBis
net.this.agent.defaultDomain=myDomain
```

in questo file viene cambiato il nome dell'agente, da MyAgentName a MyAgentNameBis. Il secondo passaggio consiste nel cambiare il file word.xml cambiando il nome dell'agente nel tag <agent>.

```
<world ver='1' >
  <domain name='myDomain' desc='This is the
                 predefined default domain.' >
    <service name='q1' serviceType='queue'
                 persistent='false' />
    <service name='t1' serviceType='topic'
                 persistent='false' />
    <service name='AuditTopic' serviceType =
             'topic' persistent='false' />
    <service name='SampleQueue2'
   serviceType='queue' persistent='false' />
    <service name='StatsTopic'
   serviceType='topic' persistent='false' />
    <service name='SampleTopic1'
   serviceType='topic' persistent='false' />
    <agent name='myAgentNameBis' >
      <transport ip='0.0.0.0' port='6677'
                 type='TCP' />
    </agent>
  </domain>
</world>
```

si noti che anche la porta nel trasporto viene cambiata da 6667 a 6677. Quando si eseguono diverse installazioni di Mantaray è indispensabile cambiare nome dell'agente, ma il numero di porta deve essere cambiato solo se le due installazioni sono in esecuzione sulla stessa macchina.

nuovo messaggio di tipo testuale ed associargli la stringa BADGE: seguita dal codice del badge letto da tastiera. Il messaggio viene poi inviato tramite la coda di invio inizializzata nel costruttore:

```
/** invia il badge al server per chiedere autorizzazione */
void verificaBadge( final String codiceBadge ) throws
                                        JMSException {
  Thread th = new Thread( new Runnable() {
    public void run() {
      TextMessage msg;
      try {
        System.out.println("verificaBadge() " +
                                        codiceBadge );
        msg = sendSession.createTextMessage();
        msg.setText( "BADGE: " + codiceBadge );
        sender.send(msg, DeliveryMode.NON_PERSISTENT,
          Message.DEFAULT_PRIORITY, MESSAGE_TTL);
        System.out.println("[send] " + msg.getText());
      } catch (JMSException e) {
        e.printStackTrace(); } } } );
  th.start();
}
```

si noti che tutte le operazioni avvengono all'interno di un nuovo thread. Questa è una scelta legata a problematiche attuali di Mantaray, che impediscono l'esecuzione dell'invio dallo stesso thread che esegue la lettura dei dati. Le prossime versioni dovrebbero risolvere questo problema rendendo superfluo l'uso dei thread.

NOTA

CODICE DEL PROGETTO

Il progetto C6 è organizzato in una serie di directory. I sorgenti sono sotto src, mentre le librerie sono sotto lib. Sotto classes ci sono le classi compilate, mentre sotto eclipse_classes quelle prodotte da Eclipse, ambiente utilizzato per lo sviluppo.

ACCESSO AI DATI

Tutte le operazioni che sono pertinenti ai dati sono contenute nella classe *BadgeDAO*. Questa è un singleton: ne esiste una sola istanza per applicazione, a cui si accede tramite il metodo *getInstance()*. Nel costruttore della classe è presente il codice di inizializzazione del driver JDBC di accesso a MySQL:

```
package it.bigatti.c6;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
public class BadgeDAO {
  Connection conn = null;
  private static BadgeDAO instance = null;
  private BadgeDAO() throws SQLException,
                        ClassNotFoundException {
    Class.forName("org.gjt.mm.mysql.Driver");
    conn = DriverManager.getConnection("jdbc:mysql:
          //10.254.120.100/c6","root","mysql");}
  public static BadgeDAO getInstance()
  throws SQLException, ClassNotFoundException {
    if( instance == null ) {
      instance = new BadgeDAO();}
  return instance;
}
```

il metodo *verificaBadge()* esegue una query sulla tabella BADGES per verificare se il badge è valido e ritorna un valore booleano che indica se il codice è corretto o meno:

```
boolean verificaBadge( String badge ) throws
                                    SQLException {
  boolean result = false;
  String sql = "SELECT id FROM badges WHERE " +
        " codice_badge='" + badge + "' AND" +
      " data_scadenza>=NOW() AND" + " valida=1";
  System.out.println( sql );
  Statement stmt = null;
  ResultSet rs = null;
  try {
    stmt = conn.createStatement();
    rs = stmt.executeQuery( sql );
    result = rs.next();
  } finally {
    if (stmt != null ) {
      stmt.close(); }
    if (rs != null ) {
```

MESSAGGI IN CODA

Per gestire la comunicazione tra il lettore di badge ed il server di controllo viene utilizzata una coda messaggi di Mantaray. Questa tecnologia permette la comunicazione punto-a-punto tra due diversi processi. In questo caso il processo client è in esecuzione sul dispositivo di lettura del badge, che si potrebbe immaginare come una periferica J2ME. Il processo server potrebbe essere in esecuzione su una macchina dedicata, che potrebbe anche avere installato il database MySQL. La comunicazione tra i due punti avviene tramite una coda: un processo invia messaggi alla coda, mentre l'altro li riceve. I messaggi vengono mantenuti nella coda nell'ordine in cui sono stati inviati, fino a che chi riceve non li legge. Una volta letti vengono eliminati. Il client di C6 utilizza una coda per inviare i codici dei badge letti al server, mentre ne utilizza un'altra per ricevere le risposte. Le due code sono BadgeIn e BadgeOut. Per poter utilizzare queste due code è necessario configurare Mantaray in modo opportuno:

1 CONFIGURAZIONE CODA CLIENT - È necessario inserire i nomi delle code nel file *word.xml*, sotto l'elemento *<domain>*, indicando il tipo di servizio (queue) e la persistenza:

```
<service name='BadgeOut' serviceType=
            'queue' persistent='false' />
<service name='BadgeIn' serviceType=
            'queue' persistent='false' />
```

2 CONFIGURAZIONE CODA SERVER - La stessa configurazione deve essere presente nel file word.xml del server. Introdurre quindi gli stessi elementi XML descritti al punto precedente nel file *word.xml* del server

3 CONFIGURAZIONE COORDINATORI - Ciascuna coda ha un coordinatore che ne gestisce i messaggi in arrivo. Un coordinatore è implementato da una istanza di Mantaray. Nel file di configurazione *default_config.params* del client compilare la proprietà coordinated_queues in questo modo:

```
coordinated_queues=BadgeIn
```

nel file del server introdurre invece:

```
coordinated_queues=BadgeOut
```

in questo modo ciascuna istanza di Mantaray gestisce una singola coda.

```
        rs.close();  } }
    return result; }
```

il metodo registra() memorizza la riga di registrazione nella tabella REGISTRAZIONE utilizzando una INSERT:

```
/** aggiunge una registrazione di ingresso */
void registra( String badge ) throws SQLException {
String sql = "INSERT INTO registrazioni "+
                    "VALUES ('"+badge+"',NOW(),'E')";
System.out.println( sql );
Statement stmt = null;
try {
    stmt = conn.createStatement();
    stmt.execute( sql );
} finally {
  if (stmt != null ) {
      stmt.close(); }}
}
```

infine, il metodo di utilità badgeCode() estrae il solo codice del badge dalla linea di testo che il client invia al server:

```
/** elimina il prefisso BADGE: dalla riga di testo ricevuta */
public static String badgeCode(String badgeLine) {
   final String prefix = "BADGE: ";
   String badge = badgeLine.substring(
          badgeLine.indexOf( prefix ) + prefix.length() );
   return badge; } }
```

CONCLUSIONI

Per provare il programma è sufficiente lanciare, in due finestre separate, i batch client.bat e server.bat. Digitando un qualsiasi codice nella finestra client, questo viene inviato al server, che gli ritorna una risposta. Attenzione: è necessario che sulla stessa macchina sia attivo un database MySQL con un database chiamato c6 e che contenga le tabelle descritte all'inizio dell'articolo.

Massimiliano Bigatti

ATTENZIONE ALLA VERSIONE!

Con Mantaray è importante utilizzare una versione aggiornata di Java. La versione 1.4.2_06 è il minimo richiesto per funzionare.

CREAZIONE DEL SERVER

La classe BadgeServer, che implementa il server dell'applicazione C6, è dotato di un costruttore che esegue le stesse inizializzazioni del client, con la sola differenza che utilizza i nomi delle code invertiti. È cioè in lettura sulla coda dove il client è in scrittura.
Il server è in scrittura sulla codea dove il client è in lettura.

```
//Crea la coda di invio
javax.jms.Queue sendQueue =
            sendSession.createQueue(
     C6Constants.RECEIVE_QUEUE_NAME);
sender = sendSession.createSender(
                          sendQueue);
//Crea la coda di ricezione
javax.jms.Queue receiveQueue =
            receiveSession.createQueue(
     C6Constants.SEND_QUEUE_NAME);
javax.jms.QueueReceiver qReceiver =
receiveSession.createReceiver(receiveQueue);
```

anche sul server viene installato un ascoltatore dei messaggi in arrivo, ma questa volta le operazioni svolte sono differenti.

```
qReceiver.setMessageListener( new MessageListener() {
   public void onMessage(Message message) {
      final Message msg = message;
      Thread th = new Thread( new Runnable() {
          public void run() {
            TextMessage tmsg = (TextMessage)msg;
            String line;
            try { line = tmsg.getText();
                System.out.println( "[receive]" + line );
```

1 CREAZIONE DI UN THREAD - La classe che implementa il listener dei messaggi esegue le operazioni all'interno di un nuovo *thread*, per gli stessi motivi visti prima. Una volta estratto il testo del messaggio è possibile procedere.

```
                //estrae il codice badge
                String badge = BadgeDAO.badgeCode(line);

                //verifica validità badge
                String badgeEsiste = BadgeDAO.getInstance().
                verificaBadge( badge ) ? "OK" : "KO";
```

2 VERIFICA BADGE - Dalla riga di testo ricevuta viene estratto solo il codice del badge e viene verificata la validità dello stesso. Entrambe le operazioni si avvalgono della classe BadgeDAO, che verrà descritta in seguito.

```
//invia il messaggio di risposta
TextMessage
msgSend = sendSession.createTextMessage();
msgSend.setText("BADGE: " + line + " " + badgeEsiste );
sender.send( msgSend, DeliveryMode. NON_PERSISTENT,
                     Message.DEFAULT_PRIORITY, MESSAGE_TTL);
System.out.println( "[send] " + msgSend.getText() );
```

3 INVIO RISPOSTA - Una volta determinato se il badge può passare o meno, viene creato un messaggio testuale che contiene il codice del badge e l'indicazione OK/KO. Questo viene poi inviato come di consueto:

```
 if (badgeEsiste.equals("OK")) {BadgeDAO.getInstance().
    registra(badge);}
} catch (JMSException e)
{e.printStackTrace();}
catch (SQLException e)
{e.printStackTrace();}
catch (ClassNotFoundException e) {e.printStackTrace();} }} );
     th.start();}
});
```

4 MEMORIZZAZIONE E CHIUSURA - Una volta inviata la risposta, viene memorizzata una riga nella tabella REGISTRAZIONI. Il metodo termina con il controllo delle eccezioni e l'avvio del thread.

Accesso a risorse interne su siti Web pubblici

Una URL, tanti documenti... personalizzati!

Non sempre è opportuno esporre risorse, immagini o documenti, su una directory pubblica di un server Web. Usiamo dei filtri che permettano di discriminare chi e come vi può accedere

Sul web, capita spesso che alcune risorse siano davvero pubbliche, altre non lo sono ma si vogliono rendere accessibili solo a utenti registrati o in possesso di particolari requisiti; ad esempio utenti autenticati tramite certificato SSU. Spesso c'è interesse anche a conoscere chi e come utilizza certe risorse, mantenendo statistiche o conteggi particolari all'interno dell'applicazione. Una soluzione che può soddisfare tutte queste esigenze può essere quella di non rendere visibili le risorse usando una directory pubblica del server Web, ma utilizzare una *url* a cui risponde un'applicazione che si preoccupa di reperire, in una zona interna del file system, la risorsa voluta e mostrarla all'utente dopo aver effettuato le opportune elaborazioni. L'utente che esegue la richiesta non si accorge che la risorsa è reperita dinamicamente. Questo tipo di approccio offre allo sviluppatore nuove e potenti funzionalità di collezione dati, protezione delle risorse e generazione selettiva e personalizzata delle risorse richieste. Supponiamo di avere questa problematica concreta: sul file system abbiamo due tipi di documenti *PDF;* il primo è solo un abstract del documento vero e proprio, il secondo è il documento completo. Vogliamo realizzare una servlet che offra a tutti la possibilità di eseguire il download dell'*abstract;* la stessa servlet limita ai soli utenti in possesso di un certificato client *X509* la possibilità di eseguire il download del documento completo. La cosa che può risultare sorprendente è che la URL dei due tipi di documenti può essere identica!
Un altro esempio può essere riferito alla questione delle immagini. Usando la stessa servlet, possiamo concedere l'accesso a due di immagini: la prima è una versione ridotta della seconda. Anche in questo caso la prima è accessibile a tutti, la seconda solo agli utenti che presentano un certificato X509.

Fig. 1: *Il mapping eseguito dalla servlet*

CREARE LA SERVLET CHE ESEGUE IL "MAPPING"

Per prima cosa creiamo una servlet che, quando risponde ad una richiesta, riconosce la url con cui è stata invocata. Da tale *url* si ricava un percorso e un nome di file: il percorso permette di discriminare parti diverse del file system, il nome del file è lo stesso che si assume presente sul file system.
Per esempio, la *url*:

http://mioserver.it/applicazione/path1/path2/immagine.gif

potrebbe essere un'applicazione Web Java di nome *"applicazione";* il percorso è *"path1/path2",* mentre il nome di file è *"immagine.gif".* La servlet, una volta mappato il percorso su una directory, che può essere *c:\immagini*, legge il file (in questo caso *c:\immagini\immagine.gif*) e lo serializza come risposta verso il client. Il metodo da implementare è il *doGet* che risponde a interrogazioni di tipo *get* (ma potremmo anche redirigere il *doPost* verso di esso, in maniera da realizzare indifferentemente le due modalità implementandone solo una):

```
public void doPost(HttpServletRequest request,
                              HttpServletResponse response)
  throws ServletException, IOException {
    doGet(request, response);}
public void doGet(HttpServletRequest request,
                              HttpServletResponse response)
  throws ServletException, IOException {
    // da realizzare }
```

Per prima cosa è necessario capire come la servlet è

stata invocata: in pratica si assume che l'URL di invocazione contenga tutti i dati per poter rispondere in maniera adeguata alla richiesta. Anziché scrivere tutto all'interno della servlet, useremo delle classi per implementare le funzionalità ausiliarie. Questo ci permette di mantenere molto semplice la servlet e di modificare le classi ausiliarie per scopi specifici senza per questo alterare la logica della servlet.
Chiameremo la prima classe *utils;* conterài due metodi che, a partire dall'url di invocazione (reperita dall'oggetto request) della servlet, reperiscono il path e il nome del file:

```
public static String getUrlPath(HttpServletRequest r) {
  String uri = r.getRequestURI();
  uri = uri.substring(r.getContextPath().length()+1);
  int i = uri.lastIndexOf("/");
  if (i==-1) return "";
  else return uri.substring(0, i);}
public static String getFilename(HttpServletRequest r) {
   String uri = r.getRequestURI();
   int i = uri.lastIndexOf("/");
   if (i==-1) return uri;
   else return uri.substring(i+1);
}
```

Da qualche parte possiamo avere un'associazione *(mapping)* tra quanto richiesto e una zona di file system. Queste informazioni potrebbero essere memorizzate in una *Hashtable* e rese accessibili da un'altra classe, che chiamiamo mapper:

```
public class mapper {
  Hashtable real;
  public String getReal(String path) {
    return (String) real.get(path);}
}
```

All'interno della stessa classe possiamo creare un metodo che, dato il nome di un file o solo la sua estensione, restituisce il suo *mime-type:*

```
public String getMimeType(String file) {
  if (file==null) return null;
  int pt = file.indexOf(".");
  if (pt>-1) file=file.substring(pt+1);
  return (String) mime.get(file.toLowerCase());}
```

Ovviamente il costruttore della classe si preoccupa di riempire le due hashtable con gli opportuni valori:

```
public mapper() {
  real = new Hashtable();
  real.put("img", "c:\\immagini\\");
  real.put("pdf", "c:\\documenti\\");
  mime = new Hashtable();
  mime.put("pdf", "application/pdf");
  mime.put("xls", "application/vnd.ms-excel");
  mime.put("doc", "application/msword");
  mime.put("gif", "image/gif");
  mime.put("jpg", "image/jpeg");
  mime.put("jpeg", "image/jpeg");
  mime.put("htm", "text/html");
  mime.put("html", "text/html");
  mime.put("txt", "text/plain");
}
```

Anziché assegnare questi valori da codice, è possibile modificare il costruttore e far sì che li carichi da una tabella di database, rendendo maggiormente configurabile l'applicazione. Si noti che questa modifica non toccherebbe minimamente l'interfaccia dei metodi esposti dalla classe né la logica della servlet che li utilizza. Non resta che creare un metodo che, dato un oggetto di tipo *File*, lo serializza sull'oggetto *response* della servlet; anche questo metodo farà parte della classe *utils:*

```
public static void outputFile(File file,
                HttpServletResponse r) throws Exception {
  r.setContentLength((int) file.length());
  FileInputStream in = new FileInputStream(file);
  OutputStream out = r.getOutputStream();
  byte[] buf = new byte[1024];
  int count = 0;
  while ((count = in.read(buf)) >= 0){
    out.write(buf, 0, count);}
  in.close();
  out.close();
}
```

Il metodo *doGet* potrebbe eseguire le seguenti operazioni:

```
String urlPath = utils.getUrlPath(request);
String filename = utils.getFilename(request);
String realPath = mp.getReal(urlPath);
String mimeType = mp.getMimeType(filename);
File file = new File(realPath+filename);
utils.outputFile(file, response );
```

Questo realizza quanto ci siamo proposti in prima istanza: eseguire il mapping tra una URL e una zona del file system e presentare il file all'utente.

PREPARARE IL DEPLOY

Creata la servlet non resta che compilarla e prepararne il deploy verso un *Servlet Container.* Scegliamo Tomcat versione 5, ma si potrebbe scegliere un qualunque altro container. La prima cosa è creare la struttura dell'applicazione Web, comprensiva di file *web.xml* che la descrive. In esso scriviamo anche i mapping della servlet verso diverse URL che voglia-

X509
Lo standard X509 definisce i dati che un certificato deve contenere. La versione 3, l'ultima, permette anche di definire estensioni ovvero dati specifici al contesto applicativo.

PDF
Per generare file PDF esistono i prodotti della Adobe; però quasi tutti i Word Processor possiedono funzionalità (standard o realizzate attraverso plug-in) per salvare i documenti in questo formato. Si veda, in particolare, Open Office alla pagina http://www.openoffice.org

I MIME TYPE
Quando un client (browser) richiede una risorsa, oltre al contenuto il server risponde con un mime type (o Internet Media Type): questo indica al client come trattarlo. In particolare può visualizzare il file (è il caso di immagini standard o file html) oppure può demandare ad altri programmi la sua visualizzazione (come file PDF, documenti Word e così via). Il registro dei mime type è presente alla pagina http://www.iana.org/assignments/media-types/

mo rispondano. Proviamo a testare la servlet avendo cura di specificare una URL con un mapping esistente e con un nome di file che esiste.

POSSIBILI ECCEZIONI

Ora possiamo capire quando ci possono essere delle eccezioni nell'invocazione della servlet e decidere come intervenire; questa fase è sempre necessaria quando l'applicazione può essere utilizzata da utenti finali: gli errori sono sempre possibili, ma è necessario prevederli e dare delle risposte significative.

Ecco alcune eccezioni che si possono verificare:

Fig. 2: *Le directory con i file per il deploy*

1. non esiste mapping tra il *path* richiesto e un *path* sul file system;
2. non esiste il file richiesto;
3. l'estensione del file non è tra i *mime-type* conosciuti;

Fig. 3: *Il risultato dell'invocazione della servlet*

è necessario prevedere come l'applicazione risponde in questi casi. Per esempio è possibile generare un documento PDF di istruzioni su come si accede al sistema o, in alternativa, generare una pagina HTML di errore o, ancora, settare degli errori standard sul response. In quest'ultimo caso possiamo inserire i seguenti controlli:

```
String mimeType = mp.getMimeType(filename);
if (mimeType == null) {
  System.out.println("Non esiste mimetype di "+filename);
  response.setStatus(HttpServletResponse.SC_
                                INTERNAL_SERVER_ERROR);
  return;
}
```

In pratica la mancanza del mimetype è considerata un errore interno. Inoltre ecco la verifica che il file esista e non sia vuoto:

```
File file = new File(realPath+filename);
if(!file.exists() || file.length()==0){
  System.out.println("Il file non esiste o ha lunghezza =0");
  response.setStatus(
                HttpServletResponse.SC_BAD_REQUEST );
  return;
}
```

In quest'ultimo caso si dà all'utente l'indicazione che la richiesta non è ben formata. Ovviamente potrebbero esserci delle altre eccezioni, come impossibilità ad accedere al file (per esempio perché non si hanno i diritti di lettura) o perché l'hard disk si è corrotto; queste eccezioni però sono dovute a errori "concettuali" (mancanza di diritti) o non prevedibili. Pertanto o si risolvono a monte (dando opportuni diritti) o si segnalano l'eccezione e le possibili cause direttamente all'utente e, magari, si intraprendono azioni automatiche di segnalazione all'amministratore del sistema.

RICONOSCERE L'UTENTE

Visto che la funzionalità di base della servlet è stata realizzata e verificata, vediamo come variare la sua risposta in base all'autenticazione dell'utente che presenta un certificato X509.

Nel numero 87 della rivista è stato visto come configurare Tomcat al fine di permettere l'accesso con certificato via HTTPS. Si ricorda che per configurare il server è necessario reperire un certificato valido o se ne genera uno (certificato *self-signed*); poi si configura il Web Server per utilizzare SSL (modifica del file server.xml impostando un opportuno connettore). Assumendo che Tomcat sia configurato opportunamente, possiamo reperire il vettore di certificati con:

```
Certs = (java.security.cert.X509Certificate[])
  request.getAttribute(
            "javax.servlet.request.X509Certificate"));
```

possiamo modificare la classe *mapper* affinché il metodo *getReal* accetti anche questo parametro:

```
String realPath = mp.getReal(urlPath, Certs);
```

Ecco come potremmo modificare tale classe:

```
public String getReal(String path) {
  return getReal(path, null);
}
public String getReal(String path, X509Certificate[]
                                              cert) {
  if (cert==null || cert.length==0)
    // nessun certificato
    return (String) real.get(path);
  else
    // esiste certificato
    return (String) realCertified.get(path);
}
```

GLOSSARIO

DESIGN PATTERN

Un pattern, o più precisamente un "Design Pattern", è la formalizzazione di una soluzione generica (quindi non legata ad un ambito particolare) applicabile ad un problema ricorrente.

Dove *realCertified* è una nuova Hashtable. Non resta che dichiarare (e inizializzare) opportunamente questa nuova hashtable per poi avere un comportamento diversificato nel caso l'utente presenti un certificato valido oppure no. Ecco come possiamo inizializzare la nuova Hashtable:

```
realCertified = new java.util.Hashtable();
realCertified.put("img", "c:\\immagini\\completi\\");
realCertified.put("pdf", "c:\\documenti\\completi\\");
```

ora i documenti nella directory *c:\documenti* saranno accessibili a tutti, come pure le immagini sotto *c:\immagini*, invocando la servlet con, rispettivamente */pdf/* e */img/;* se si invoca la stessa servlet con un certificato verranno mostrati i documenti presenti in *c:\documenti\completi* e le immagini presenti in *c:\immagini\completi*.

COSA MANCA?

L'applicazione dell'utente manca di alcune cose fondamentali in qualunque ambiente di produzione: la configurabilità del mapping e il censimento al run-time degli utenti abilitati al servizio (autorizzazione, successiva all'autenticazione); probabilmente manca anche un conteggio delle risorse consumate e di statistiche di utilizzo per ogni utente o per ogni risorsa. Tutte queste problematiche potrebbero essere risolte utilizzando un DBMS; a tal proposito si possono realizzare dei Java bean che implementano le funzionalità di accesso e memorizzazione dei dati da e verso i database usando la tecnologia JDBC.

Fig. 4: ***Il documento completo, acceduto alla stessa URL ma usando un certificato X509***

In ambito J2EE quasi tutte le moderne applicazioni Web seguono il pattern MVC, acronimo delle tre funzionalità base: *Model, View, Controller.* Si tratta di realizzare, attraverso componenti separati, queste funzionalità. L'applicazione di esempio ha come controller la servler, in quanto si preoccupa di prendere la richiesta e "controllare" chi e come la deve soddisfare; il model è dato dai Java Bean che hanno, in questo caso, la logica applicativa (come effettuare il mapping) ma mancherebbe la view, ovvero un componente separato per effettuare la visualizzazione dei dati. È vero che, in parte, è il JavaBean utils che la realizza però, di solito, sempre in ambito J2EE, sono le pagine *JSP (JavaServer Pages)* che si occupano di mostrare i risultati. Nel nostro esempio aver "sacrificato" la parte di view è giustificato dal fatto che la visualizzazione è una semplice lettura e scrittura di array di byte.

IN PROSPETTIVA

Si è visto come una servlet che esegue il mapping dinamico da un'URL a una risorsa interna possa selezionare sia il tipo di accesso ma anche il tipo di risultato in base a fattori propri dell'applicazione. La stessa logica è utilizzata, per esempio, da quelle applicazioni che fungono da contatori per gli accessi ai siti Web e che, come risultato, mostrano un'immagine che rappresenta, numericamente, il numero di visitatori. Una cosa interessante è anche quella di generare i contenuti a partire dal file reale (per esempio un documento o un'immagine completa), ma manipolandoli in maniera da includere il nome del richiedente (reperito dal certificato) al suo interno. In questo modo il contenuto non solo viene mostrato in maniera diversa tra utenti non autenticati e utenti con certificato, ma personalizzato per ognuno rendendo "rintracciabile" a posteriori chi ha eseguito il download!
Ovviamente gli utilizzi possono essere tra i più disparati. Proprio per questo il consiglio è quello di partire dall'esempio proposto e di personalizzarlo per i propri scopi ma senza perdere di generalità dell'applicazione, in maniera da poterla mantenere e far evolvere come prodotto unico per più applicazioni.

Ivan Venuti

Fig. 5: ***Il pattern MVC***

JDBC

JDBC è lo standard di accesso alle basi dati di Java. Grazie all'uso di driver dalle funzionalità standard, un'applicazione scritta per uno specifico DBMS è portabile su qualsiasi altro che abbia, a sua volta, driver JDBC.

• **TECNICHE DI AUTENTICAZIONE ATTRAVERSO LA GESTIONE DI CERTIFICATI X509** ***I. Venuti*** **(ioProgrammo n.??) 2005**

• **JAVA SERVLET PROGRAMMING Seconda Edizione** ***J. Hunter*** **(O'Reilly) 2001**

GPS per PC: determiniamo posizione, quota e percorso compiuto

Il mio primo GPS

In questo articolo imparerete ad estrarre posizione, quota ed orario da un ricevitore GPS, a visualizzare la traccia dello spostamento e a interfacciare lo strumento con il PC

I TUOI APPUNTI

Utilizza questo spazio per le tue annotazioni

REQUISITI

Conoscenze richieste
Programmazione Delphi, concetti base di elettronica

Software
S.O. Win 9.x, ME, 2000, NT, XP

Impegno

Tempo di realizzazione

Probabilmente molti lettori hanno a disposizione un ricevitore GPS che vogliono impiegare in modo completo, desiderando di sfruttarne tutte le potenzialità andando oltre quanto offerto dal costruttore, sviluppando applicazioni proprie. In linea generale quasi ogni GPS dispone di un connettore esterno atto ad essere collegato ad un'alimentazione, come ad esempio per l'impiego in auto od in barca, oltre che di un connettore di I/O, generalmente seriale. Sfrutteremo appunto la porta seriale per connettere il GPS al PC.

IL CODICE

Per seguire agevolmente quanto descriveremo nell'articolo è indispensabile installare il programma spuntosoft allegato ad ioProgrammo. Realizzeremo inoltre un programma in Delphi che cattura i dati trasmessi via seriale dal GPS e assegna a video la traiettoria dello spostamento. La unit principale contiene praticamente tutto il codice, ad eccezione della form contenente la licenza d'uso ed il doveroso 'disclaimer', oltre che dall'ottimo componente freeware *'CPDrv.pas'* che può essere scaricato gratuitamente dal Web *(Cdd4.zip)* del quale si allegano le condizioni di utilizzo in tutte le copie del programma, come prescritto dal relativo autore. Il componente visuale *'SpuntoHyperLabel'* che consente il collegamento ipertestuale di una qualunque etichetta inserita nella form, viene fornito nella directory di installazione del programma.

```
unit GPS_Position_Plotter_Main;
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics,
                    Controls, Forms, Dialogs,
  Menus, ComCtrls, ExtCtrls, StdCtrls, ToolWin,
                    ImgList, ClipBrd, CPDrv,
  GPSPositionPlotterAboutUnit, SpuntoHyperLabel, jpeg;
```

Di seguito viene riportato un estratto della classe principale del programma, che è stata debitamente semplificata per motivi di leggibilità. In particolare si notano le poche procedure che sono deputate alla gestione del programma, tra le quali figurano quelle deputate alla connessione e disconnessione con il GPS, all'invio di comandi NMEA al GPS, alla ricezione dati ed alla visualizzazione delle coordinate, nonché alla rappresentazione grafica del percorso.

```
TGPSPositionPlotterMainForm = class(TForm)
   ....................................
  procedure ConnectClick(Sender: TObject);
  procedure DisconnectClick(Sender: TObject);
  procedure NMEATextOutputKeyPress(Sender:
                    TObject; var PressedKey: Char);
  procedure SerialDriverReceiveData(Sender:
        TObject; DataPtr: Pointer; DataSize: Cardinal);
  procedure Quit(Sender: TObject);
  procedure FormCreate(Sender: TObject);
  procedure HelpMenuClick(Sender: TObject);
  procedure NMEATextInputChange(Sender: TObject);
  procedure NMEApresentPosition;
  procedure FormResize(Sender: TObject);
 private
 public
end;
```

Di seguito si riportano le variabili globali del programma, che faciliteranno al lettore la comprensione delle procedure seguenti.

```
var
 GPSPositionPlotterMainForm: TGPSPositionPlotterMainForm;
 GlobalString,OldGGAStringGlobal:String;
 GlobalRetrivedNMEAString:integer;
 InitialPositionLat,InitialPositionLon,MapScale:Real;
 Startup:Boolean;
```

Al momento della creazione della form principale viene eseguita la procedura *FormCreate*, che ha lo scopo di visualizzare la finestra iniziale contenente licenza e disclaimer, di inizializzare le variabili di posizione del sistema ed infine predisporre la forma grafica della form, rappresentante lo scostamento dalla posizione iniziale del ricevitore GPS in forma

radiale. Allo scopo di facilitare la visualizzazione del movimento del GPS vengono rappresentati una sequenza di cerchi concentrici distanti l'equivalente di diciotto metri di spostamento: il lettore potrà variare questa misura in funzione delle proprie esigenze di visualizzazione.

```
procedure TGPSPositionPlotterMainForm.FormCreate(
                                        Sender: TObject);
var
GPSPositionPlotterAboutdlg: TGPSPositionPlotterAboutBox;
I,Xc,Yc:Integer;
begin
 // About Dialog
 GPSPositionPlotterAboutdlg := nil;
 GPSPositionPlotterAboutdlg :=
TGPSPositionPlotterAboutBox.Create( Self );
 GPSPositionPlotterAboutdlg.ShowModal;
 // Initial Position
 InitialPositionLat:=45;
 InitialPositionLon:=15;
 Startup:=True;
 //Graphic Map
 Mapscale:=1000;   //1 pixel equal to 1,8 mt
 MapImage.Picture.LoadFromFile('800x600blank.BMP');
 MapImage.Canvas.Pen.Color:=clYellow;
 MapImage.Canvas.Brush.Style:=bsClear ;
 xc:=( MAPPanel.Width div 2);
 yc:=(MAPPanel.Height div 2);
 For I:=0 to 8 do begin
   MapImage.Canvas.Ellipse((Xc-(I*50)),(Yc-(I*50)),
                          (Xc+(I*50)),(Yc+(I*50)));
 end;
end;
```

In seguito alla attivazione dell'item *'About and licence'* sulla form principale, viene eseguito il codice che segue, che provvede a visualizzare la form relativa alla licenza ed al disclaimer di questo programma.

```
procedure TGPSPositionPlotterMainForm
                      .HelpMenuClick(Sender: TObject);
var GPSPositionPlotterAboutdlg: TGPSPositionPlotterAboutBox;
begin
 GPSPositionPlotterAboutdlg := nil;
 GPSPositionPlotterAboutdlg :=
            TGPSPositionPlotterAboutBox.Create( Self );
 GPSPositionPlotterAboutdlg.ShowModal;
end;
```

La connessione e la disconnessione del PC dal ricevitore GPS, avviene in seguito all'attivazione degli items *'GPS/Connect to GPS'* e *'GPS/Disconnect from GPS'* e dell'esecuzione delle procedure corrispondenti, riportate di seguito, dal funzionamento abbastanza ovvio ed intuitivo.

```
procedure TGPSPositionPlotterMainForm.ConnectClick(
                                        Sender: TObject);
begin
 SerialDriver.Connect;          // Connection to GPS
end;
procedure TGPSPositionPlotterMainForm
                     .DisconnectClick(Sender: TObject);
begin
 SerialDriver.Disconnect;    // Disconnection from GPS
end;
```

Nel caso sia necessario inviare comandi al modulo GPS, è possibile utilizzare manualmente l'apposita casella di edit (*'NMEA Output Command'*), oppure utilizzare la procedura che segue per l'invio di un singolo carattere al ricevitore attraverso la porta seriale. La procedura in questione infatti non è altro che l'event handler dell'evento *'OnKeyPress'* del componente *'NMEATextOutput'*.

```
procedure TGPSPositionPlotterMainForm
           .NMEATextOutputKeyPress(Sender: TObject;
                              var PressedKey: Char);
begin
  if not SerialDriver.Connected then
    exit;
  SerialDriver.SendChar( PressedKey );
end;
```

Nell'eventualità in cui la finestra venga ridimensionata, il codice riportato di seguito provvede a ricaricare lo sfondo della form principale ed a ridisegnare I cerchi concentrici corrispondenti allo spostamento del ricevitore dalla posizione iniziale: ovviamente la finestra viene cancellata e ridisegnata durante quest'operazione.

```
procedure TGPSPositionPlotterMainForm.FormResize(
                                        Sender: TObject);
Var
I,Xc,Yc:Integer;
begin
 MapImage.Picture.LoadFromFile('800x600blank.BMP');
   MapImage.Canvas.Pen.Color:=clYellow;
   MapImage.Canvas.Brush.Style:=bsClear ;
   xc:=( MAPPanel.Width div 2);
   yc:=(MAPPanel.Height div 2);
   For I:=0 to 8 do begin
     MapImage.Canvas.Ellipse((Xc-(I*50)),(
                  Yc-(I*50)),(Xc+(I*50)),(Yc+(I*50)));
   end;
end;
```

CATTURARE I DATI

Siamo arrivati finalmente al cuore del programma, dove viene riconosciuta e gestita la 'sentence' della posizione geografica GPS. Quando il driver

Il programma funziona utilizzando la porta seriale COM1, che in alcuni PC viene utilizzata per la gestione del mouse, in questo caso per evitare conflitti di sistema è necessario variare i parametri di connessione del componente 'SerialDriver'. I parametri di comunicazione della porta seriale vengono impostati come: COM1, 9600 baud, 8 bit dati, 1 bit di Stop: se sul proprio sistema si intende utilizzare una porta differente, oppure parametri diversi da quelli citati è sufficiente variare i parametri di connessione del componente 'SerialDriver' attraverso l'Object Inspetor di Delphi.

NOTA

CONNESSIONE SERIALE

Il programma funziona acquisendo i dati dal GPS dalla porta seriale COM1, 9600 baud, 8 bit dati, 1 bit di Stop, se sul proprio sistema si intende utilizzare una porta differente, oppure parametri diversi da quelli citati è sufficiente variare i parametri di connessione del componente 'SerialDriver' attraverso l'Object Inspetor di Delphi.

CONTATTA L'AUTORE

L'autore è lieto di rispondere ai quesiti dei lettori sull'interfacciamento dei PC all'indirizzo: luca.spuntoni@ioprogrammo.it

'SerialDriver' riceve una stringa di caratteri dal GPS attraverso la porta seriale, provvede ad aggiornare la casella di testo *'NMEATextInput'*. Questo evento innesca l'esecuzione dell'event handler seguente che a sua volta provvede a verificare se ci si trovi in presenza di una stringa *'$GPGLL' (Geographic Position, Latitude/Longitude)*, che ingloba tutte le informazioni che ci sono necessarie. In effetti questa frase è probabilmente la più significativa ai fini della definizione della posizione del ricevitore, poiché fornisce la latitudine geografica, la longitudine e l'orario al quale è stata effettuata la misurazione. Se la verifica ha risultato positivo, viene eseguita la procedura *'NMEAPresentPosition'*.

```
procedure TGPSPositionPlotterMainForm
                  .NMEATextInputChange(Sender: TObject);
var s: string;
begin
 if SerialDriver.Connected then
   begin
   s :=NMEATextInput.Lines[NMEATextInput.Lines.Count-2];
   If (copy(S,1,6)='$GPGGA') and (s<>
                           OldGGAStringGlobal) then begin
          OldGGAStringGlobal:=s;
          Globalstring:=s;
          NMEApresentPosition;
        end;
       end;
```

La procedura '*NMEAPresentPosition*' provvede ad estrarre le coordinate geografiche, l'ora e la quota del ricevitore: il formato riferito al protocollo NMEA per la precisione è il seguente:

```
Latitude ddmm.mmmm
N/S Indicator
Longitude dddmm.mmmm
E/W indicator
UTC Time hhmmss.sss
Status character A A=data valid V=data invalid
Checksum
<CR> <LF>
```

PRECAUZIONI

Prima di collegare il circuito al nostro PC occorre verificare la nostra realizzazione con attenzione per assicurarci che tutto sia stato collegato come previsto. Si raccomanda di verificare che il ricevitore GPS che si intende utilizzare abbia una porta di comunicazione compatibile con quella del Personal Computer, verificando le rispettive documentazioni tecniche. Il programma funziona utilizzando la porta seriale COM1, che in alcuni PC viene utilizzata per la gestione del mouse, in questo caso per evitare conflitti di sistema è necessario variare i parametri di connessione del componente 'SerialDriver' attraverso l'Object Inspector di Delphi, selezionando una porta seriale libera.

Nella stesura della procedura si è voluto favorire la semplicità di comprensione del codice, rispetto alla sua ottimizzazione.
Innanzi tutto si procede alla decodifica della stringa NMEA secondo il protocollo appena descritto, vengono estratti i valori corrispondenti ai vari frammenti di informazione, che vengono poi memorizzati nelle rispettive variabili. La validità dell'informazione viene confermata dalla presenza di un valore non nullo posto tra due virgole consecutive della stringa. Il ricevitore GPS, infatti, provvede a riempire i campi delle informazioni, soltanto se il valore corrispondente è realmente corretto, in generale in corrispondenza di un buon funzionamento (*tracking*) dell'apparato. Per quanto riguarda la rappresentazione grafica del percorso compiuto dal ricevitore, si provvede innanzi tutto ad estrarre la prima posizione valida fornita dal GPS, per determinare il centro dello schermo, dopo di ciò si determinano le posizioni relative a questa coordinata, che vengono poi disegnate sullo schermo.
Ovviamente il percorso che ne risulta è relativo alla posizione iniziale e non misurato in senso assoluto, ma può già essere molto utile in diverse applicazioni pratiche.

```
procedure TGPSPositionPlotterMainForm.NMEApresentPosition;
 var
  stringline,s:string[255];
  io,fo,ila,fla,ilo,flo,ialt,falt,v,m,code,X,Y:Integer;
  ora,lat,ns,lon,ew,quota,fm:string;
  feasible:Boolean;
  gra,pri:real;
  LatReal,LonReal,Quotareal:Real;
begin
  Stringline:=GlobalString;
   begin
     (*Decodifico il formato NMEA GPGGA Global Position*)
     If copy(Stringline,1,6)='$GPGGA' then begin
       io:=0; fo:=0; ila:=0; fla:=0; ilo:=0; flo:=0;
       ialt:=0; falt:=0; v:=0; ora:='';     lat:=''; lon:='';
       ns:=''; quota:=''; m:=''; Feasible:=True;
       for m:=1 to (length(stringline)-7) do begin
         If copy(Stringline,m,1)=',' then v:=v+1;
         if v=0 then io:=m;   if v=1 then fo:=m;
                                     if v=1 then ila:=m;
         if v=2 then fla:=m;  if v=3 then ilo:=m;
                                     if v=4 then flo:=m;
         if v=8 then ialt:=m;  if v=9 then falt:=m;
         (* Controllo se il GPS sta tracciando oppure no*)
         if fo>(io+1) then begin
           ora:=copy(Stringline,(io+2),(fo-io-1));
           Feasible:=True;
         end
         else
         begin
           Feasible:=False;
         end;
```

```
        if fla>(ila+1) then begin
          lat:=copy(Stringline,(ila+2),(fla-ila-1));
          Val( Lat,LatReal,code);
          ns:=copy(Stringline,(fla+2),1);
          Feasible:=True;
        end
        else
        begin
          Feasible:=False;
        end;
        if flo>(ilo+1) then begin
          lon:=copy(Stringline,(ilo+2),(flo-ilo-1));
          Val( Lon,LonReal,code);
          ew:=copy(Stringline,(flo+2),1);
          Feasible:=True;
        end
        else
        begin
          Feasible:=False;
        end;
        if falt>(ialt+1) then begin
          quota:=copy(Stringline,(ialt+2),(falt-ialt-1));
          fm:=copy(Stringline,(falt+2),1);
          Feasible:=True;
        end
        else
        begin
          Feasible:=False;
        end;
      end;
      if feasible then begin
         inc(GlobalRetrivedNMEAString);
         s:='NMEA Global Position and Altitude at: '+ora+'
           '+lat+' '+ns+' '+lon+' '+ew+' '+quota+' '+fm;
         label3.caption:=ora;   label4.caption:=lat;
                                 label9.caption:=ns;
        label8.caption:=lon;      label5.caption:=ew;
                                 label6.caption:=quota;
        label7.caption:=fm;
        // Graphic plot
        if (Startup and Feasible) then begin
                                     //First Coordinate
           InitialPositionLat:=LatReal; //Map Centre
           InitialPositionLon:=LonReal;
           Startup:=False;
        end;
        // Position Plot
        Y:= ( MAPPanel.Height div 2 ) +Trunc(
              (InitialPositionLat-LatReal)*MapScale);
       X:= ( MAPPanel.Width div 2 )+Trunc(
             (LonReal-InitialPositionLon)*MapScale);
       MapImage.Canvas.Pen.Color:=clRed;
       MapImage.Canvas.Brush.Style:=bsDiagCross;
       MapImage.Canvas.Ellipse((x-5),(y-5),(x+5), (y+5));
          end;
      end;
   end;
 end;
```

Il programma potrebbe essere utilizzato ad esempio, senza bisogno di alcuna modifica, per la misura di superfici, oppure per la monitorizzazione dello spostamento di una apparecchiatura mobile: sarebbe molto interessante, infatti, potere controllare in remoto un sistema mobile agricolo, in grado di arare, seminare ed irrigare un campo, oppure di controllare un sistema di sorveglianza mobile.

CONCLUSIONI

Utilizzando un ricevitore GPS su circuito stampato, come nel caso del modulo GEOGAP, oppure interfacciando un ricevitore commerciale il lettore è in grado di realizzare un sistema completo di monitorizzazione della posizione di un sistema mobile. Nel prossimo appuntamento vedremo come correlare la posizione geografica ottenuta dal GPS con un sistema cartografico e come realizzare un vero e proprio apparato di navigazione terrestre, navale od addirittura aereo.

Luca Spuntoni

SUL WEB

Il sistema proposto in queste pagine è stato realizzato e collaudato con la apparecchiatura per il collaudo e la sperimentazione di circuiti elettronici con Personal Computer 'PC EXPLORER light':

ulteriori informazioni su come si possa reperire questa apparecchiatura è possibile visitare il WEB all'indirizzo:

'http://www.pcexplorer.it'
oppure
'http://web.tiscali.it/spuntosoft/'
od infine inviare una e-mail a:
spuntosoft@tiscali.it.

UTILIZZO DEL PROGRAMMA

A questo punto siamo pronti a collegare il GPS se-guendo le connessioni illustrate in queste pagine, impostiamo il ricevitore in modo che trasmetta sulla sua porta seriale il flusso di informazioni secondo il protocollo NMEA, attraverso le apposite schermate di configurazione, dipendenti dal tipo di ricevitore satellitare. Lanciando il programma sul PC ed effettuando la connessione al GPS, tramite l'opzione 'Connect to GPS' dal menu GPS, si dovrebbe notare il flusso di informazioni NMEA nella finestra 'NMEA Raw informations'. In questa fase è probabile che se siamo al coperto, nelle caselle deputate a visualizzare l'orario, le coordinate geografiche e la quota, rimangano visualizzati dei semplici trattini, questo significa che il ricevitore non sta 'traccando' un numero sufficiente di satelliti da garantirci una posizione valida. È necessario posizionare il ricevitore in modo che possa 'vedere' uno spazio di cielo sufficiente a ricevere tre o quattro satelliti: se questa condizione si verifica, vedremo apparire le coordinate e gli altri dati di posizione nelle apposite caselle di dialogo. Se si dispone di un PC portatile, si può provare a spostarsi per verificare l'analogo spostamento della posizione geografica dal punto di vista numerico, nonché notare che il programma provvede a registrare in forma grafica il nostro movimento. Per completezza di trattazione possiamo affermare che alcuni ricevitori GPS dispongono di una funzione di simulazione, per mezzo della quale è possibile appunto simulare che il dispositivo riceva un congruo numero di satelliti e che magari si sposti secondo una data direzione e velocità: questa funzione, se presente può essere utilizzata per verificare il funzionamento della nostra applicazione, rimanendo comodamente seduti alla propria scrivania.

Come sfruttare i controls e gli eventi di ASP.NET

Percorsi diversi obiettivo comune

Il modello di pagina ASP.NET ricorda molto da vicino quello delle applicazioni Windows, perché entrambi sfruttano un modello orientato alla programmazione degli eventi, vediamo come

Utilizza questo spazio per le tue annotazioni

Nella precedente numero abbiamo dato una rapida occhiata ai fondamenti di ASP.NET, che ci serviranno da subito per cominciare a cimentarci nella costruzione di semplici applicazioni web basate su questa tecnologia. A partire da questo articolo, infatti, presenteremo alla fine di ogni argomento la creazione di un'applicazione di esempio che permetta di mettere da subito in pratica ciò che abbiamo imparato.
A tal proposito, per continuare, dobbiamo inquadrare il modello di pagina che caratterizza ASP.NET ed imparare a maneggiarne gli oggetti che la compongono.

IL MODELLO DI PAGINA

ASP.NET ha un modello di pagina che ricorda molto da vicino le applicazioni Windows, tanto che ne condivide per molti aspetti il nome. Per prima cosa, una pagina è anche detta Web Form, in contrapposizione alle Windows Forms che invece sono proprie di un'applicazione Windows. Una web form differisce, a prima vista, da una winform perché utilizza controls differenti, chiamati web controls, e perché a differenza di quest'ultima, una web form produce codice HTML. Dunque anche se l'application model di ASP.NET è differente da quello di una pagina web "normale", alla fine il risultato è sempre codice HTML. L'aver adottato questo tipo di approccio permette di sviluppare applicazioni web sfruttando le stesse funzionalità di applicazioni Windows, tra cui la gestione integrata degli eventi associati ad un oggetto della pagina, con però le prerogative di sviluppo di un'applicazione Windows, che in genere consente un tempo di sviluppo inferiore perché si basa su tool visuali. In più ASP.NET permette di gestire in maniera nativa, senza scrittura di codice complesso, l'intercettazione degli eventi associati ad un oggetto, come ad esempio il click su un pulsante o, più semplicemente, il submit di una form.
Più in generale la struttura della pagina, come ci accorgeremo utilizzando ASP.NET, è pensa-

METODI DI BASE DELLA PAGINA

**Sono agganciati attraverso HttpContext, una classe particolare istanziata da HttpRuntime, che espone verso la pagina l'accesso a querystring, form, etc.
I principali metodi sono:**

- **Response: fornisce un ponte verso la classe *HttpResponse*, che implementa le funzionalità di output verso il client.**
- **Request: si aggancia alla classe *HttpRequest*, utilizzata per recuperare informazioni dal client.**
- **Server: sfrutta *HttpServerUtility* per fornire funzionalità di uso comune alla pagina.**
- **Session: fornisce un ponte verso la classe *HttpSessionState*, che contiene le funzionalità necessarie alla gestione dello stato della sessione.**
- **Application: si aggancia ad *HttpApplicationState*, per fornire la logica di accesso allo stato dell'applicazione.**

ta per rendere più agevole il lavoro dello sviluppatore.

COME VIENE COMPILATA UNA PAGINA

Per prima cosa, quando richiediamo una pagina con estensione *.aspx*, qualora non fosse già in esecuzione, viene fatto partire il *worker process ASP.NET (ASPNET_wp.exe)*.
Quando la nostra pagina viene richiamata per la prima volta, avviene la compilazione vera e propria in MSIL. La pagina *prova.aspx* produrrà una classe, compilata in un assembly che chiameremo, per semplicità, *prova.dll*.
Prova.dll viene eseguito just in time in memoria e salvata su disco per le esecuzioni successive. Se la pagina non viene modificata, sarà utilizzata la versione già compilata, altrimenti (e questo vale anche se il server web viene riavviato) la procedura si ripete fino ad avere lo stesso risultato. Questa operazione permette di ottimizzare il tempo di esecuzione, dato che la compilazione è effettuata una sola volta. Nella compilazione nell'assembly corrispondente sono però racchiusi più passaggi:

- vengono individuate tutte le dipendenze con eventuali classi esterne;
- viene invocato il compilatore specifico per il linguaggio utilizzato nella pagina. Nel caso di Visual Basic.NET sarà *vbc.exe*, per C# *csc.exe;*
- viene generato un assembly e salvato su disco, in una directory temporanea, per essere riutilizzato.

La lentezza della prima esecuzione è ovvia, anche se in realtà non è visibile ad occhio nudo, se non sul server con carichi elevati o poca disponibilità di risorse.
In tutti gli altri casi è completamente trasparente ed indolore, ed avviene in pochi centesimi di secondo dalla richiesta.
Ovviamente le successive chiamate sfrutteranno la versione già compilata e pertanto il tempo necessario alla relativa esecuzione sarà impercettibile.

UNA PAGINA, UNA CLASSE

Alla base di tutto c'è la singola classe di cui ogni pagina è composta. Questo non vuol dire che per creare una pagina bisogna inventarsi qualcosa di complicato, perché grazie ad uno dei fondamenti dell'*Object Oriented Programming*, l'ereditarietà, possiamo sfruttare appieno una classe già pronta ed ovviamente contenuta all'interno del .NET Framework, chiamata *Page* e situtata nel namespace *System .Web.UI*. Questa classe contiene al proprio interno molta della logica necessaria a far funzionare una pagina HTML, come l'invio del codice al client, il recupero delle informazioni, etc. Questo consente di estendere le funzionalità della classe di base ed utilizzare una base comune, con funzionalità specifiche, all'interno delle nostre applicazioni.
Tra l'altro, Visual Studio .NET 2003, l'IDE di sviluppo pensata da Microsoft per ASP.NET, utilizza una modalità chiamata *code-behind,* che separa codice da contenuto, sfrutta proprio questa caratteristica per creare la pagina, in un file separato, una classe a tutti gli effetti da cui poi la pagina vera e propria eredita.

COME FUNZIONANO GLI EVENTI

Come tutti gli oggetti, anche la classe *Page* può avere degli eventi, che si verificano in seguito ad un comportamento del client e che possono essere di natura automatica o asso-

NOTA

IL POSTBACK ED IL VIEWSTATE

Per scatenare il *PostBack*, ASP.NET utilizza due campi nascosti all'interno della form che automaticamente aggiunge alle pagine. Si tratta di *__EVENTTARGET* e *__EVENTARGUMENT* che indicano rispettivamente il controllo che ha scatenato il *PostBack* e gli argomenti da passare. Queste informazioni serviranno ad ASP.NET per ricostruire l'evento scatenato dall'utente associato al control.
Il *ViewState* invece salva lo stato dei controls all'interno di un campo nascosto chiamato *__VIEWSTATE*

GLI EVENTI DI UNA PAGINA

Questi sono gli eventi principali che vengono scatenati all'interno di una pagina, nell'ordine in cui vengono intercettati:

- **PAGE_INIT: all'inizializzazione della classe che compone la pagina;**
- **PAGE_LOAD: al caricamento della pagina e dei controls contenuti;**
- **PAGE_PRERENDER: prima del rendering di pagine e controls.**
- **PAGE_RENDER: nella fase di rendering della pagina e dei controls.**

C'è poi un evento particolarmente interessante, *Page_Error,* che si verifica in una circostanza ben specifica, ovvero in caso di errore non gestito all'interno della pagina e che può essere sfruttato per visualizzare un avviso all'utente o registrare la causa del fallimento dell'esecuzione.

ciati a particolari condizioni. Come è possibile avere eventi in un'applicazione web, che è senza stato? Dopo tutto una pagina web comincia la sua vita non appena il client la richiede, per morire quando il suo risultato è inviato al client. ASP.NET sfrutta questo semplice concetto: tutti gli eventi si verificano sul server, ma sono invocati dal client. ASP.NET non fa altro che rendere più semplice, per chi scrive codice, pensare ad eventi, sfruttando però del normale codice Javascript per fare il submit di una form, passando i dati necessari a far eseguire il codice server side. Da parte nostra, questo è il bello, non è necessario scrivere codice Javascript, perché la pagina è in grado di fornirci la struttura di base sulla quale operare e che si occuperà di fornirci il substrato necessario all'invocazione degli eventi. Ci basterà dunque solamente definire i vari eventi e fare in modo che vengano invocati.

GLI OGGETTI DI UNA WEB FORM: I CONTROLS

Se la web form è il contenitore, gli oggetti contenuti all'interno di quest'ultima sono co-

UNA SEMPLICE PAGINA IN 5 PASSI

1 **Facciamo partire VS.NET o il nostro editor preferito. Creiamo un nuovo progetto.**

2 **Inseriamo all'interno della web form un controllo textbox, un button,, un panel ed all'interno di quest'ultimo una label.**

```
<%@ Page language="c#" Codebehind=
        "WebForm1.aspx.cs" AutoEventWireup="false"
            Inherits="ioprogrammo.WebForm1" %>
<form id="Form1" method="post" runat="server">
<P>Inserisci il tuo nome:
<asp:TextBox id="txtNome" runat="server">
                        </asp:TextBox></P>
<P>
<asp:Button id="submit" runat="server"
            Text="Invia!"></asp:Button></P>
<asp:panel runat="server" id="Panel1"
                            Visible="False">
<P>Hai detto di chiamarti:
<asp:Label id="nome" runat="server">
                    Label</asp:Label></P>
</asp:Panel>
</form>
```

3 **Il codice HTML generato automaticamente sarà simile a questo.**

```
private void Page_Load(object sender,
                        System.EventArgs e)
{
  // solo sul PostBack
  if (Page.IsPostBack)
  {
    // prelevo il testo dalla textbox
    nome.Text = txtNome.Text;
    Panel1.Visible = true;
  }
}
```

4 **All'interno del codice (nel nostro caso C#) della pagina, inseriamo l'intercettazione del postback (che nel nostro caso equivale al submit della form) ed aggiungiamo nell'evento *Page_Load* il codice necessario a recuperare dalla proprietà *Text* del controllo textbox *txtNome* il valore inserito ed associarlo alla label.**

munemente chiamati *controls*. Ovviamente producono, come output, codice HTML, anche se la loro dichiarazione all'interno della pagina ha una sintassi differente da quello che potrebbe essere un tipico tag. Ci sono due grandi famiglie di controls, divisi in base alle funzionalità che implementano:

- **Web controls:** si riconoscono perché hanno il prefisso *<asp:>;*
- **HTML controls:** sono normali tag HTML con l'attributo *runat="server"*.

La differenza principale tra queste due famiglie è che i primi ricordano, nel nome, i controls per Windows e si inseriscono all'interno delle pagine con una sintassi diversa dal corrispondente codice HTML generato, mentre i secondi permettono di utilizzare un tag HTML sfruttandone la possibilità di essere trattato come oggetto e quindi di accedere in maniera programmatica alle sue funzionalità. I web controls offrono un insieme coerente di proprietà, in modo che la stessa funzionalità implementata su controls differenti mantenga lo stesso nome all'interno di tutto il framework. In realtà ci sono altre divisioni in base alla funzionalità che i controls offrono, ma per cominciare ci basta sapere questa differenza. L'uso di una famiglia anziché di un'altra è solo questione di comodità. Molto spesso, infatti, si utilizzano controls di entrambe le tipologie insieme nella stessa pagina senza problemi.

5 **Il risultato è visibile nelle immagini qui sopra.**

IL POSTBACK ED IL VIEWSTATE

Con altre tecnologie, come Classic ASP, ogni pagina ha una propria vita e, molto spesso, se l'operazione richiede l'utilizzo di più passi logici, si opta per realizzare pagine distinte, slegate tra loro. Con ASP.NET, ad esempio, un modulo di contatto è costruito all'interno di una sola pagina, anziché crearne una dedicata alla raccolta ed un'altra alla visualizzazione del risultato. Per controllare tutti questi passaggi tra client e server viene utilizzato un meccanismo particolare che è chiamato PostBack, a cui è associato un omonimo stato che viene sfruttato, dalla pagina, per tenere traccia di questo cambio di stato. Il PostBack si verifica esattamente ogni volta che il client invia il controllo al server al seguito di un'azione compiuta dall'utente sulla pagina. È questo il motivo per cui con ASP .NET si fa pochissimo uso dell'oggetto Request, dato che si può direttamente accedere alla proprietà che contiene il testo del controllo. La pagina poi offre ai controls una specie servizio, il ViewState, che non è altro che un raccoglitore dello stato degli oggetti della stessa. In questo modo tra un PostBack ed un altro i dati vengono salvati e recuperati da questo contenitore, che è implementato come campo hidden dalla web form. Se ad esempio implementiamo un sistema di validazione dell'input dell'utente, non dovremo preoccuparci di mantenere il valore dei campi presenti sulla pagina, perché il ViewState se ne occupa direttamente per noi, risparmiandoci un'enorme fatica! Più in generale impareremo a sfruttare meglio gli eventi associati ad ogni controls (e quindi il ViewState) nel corso dei prossimi articoli.

CONCLUSIONI

Comprendere a fondo il meccanismo alla base di ASP.NET consente di sviluppare applicazioni web basate su questa tecnologia in maniera più semplice. L'utilizzo degli eventi e degli oggetti all'interno della pagina consente infatti di rendere estremamente flessibile la stessa e facilmente programmabile un oggetto in base alle nostre esigenze.

Daniele Bochicchio

L'AUTORE

Daniele Bochicchio è il content manager di *ASPItalia.com*, community che si occupa di ASP.NET, Classic ASP e Windows Server System. Il suo lavoro è principalmente di consulenza e formazione, specie su ASP.NET, e scrive per diverse riviste e siti. È Microsoft ASP.NET MVP, un riconoscimento per il suo impegno a supporto delle community e per l'esperienza maturata negli anni. Il suo blog è all'indirizzo http://blogs.aspitalia.com/daniele/

All'interno della programmazione ad oggetti

ActionScript 2.0 Metodi e Proprietà

Lavorare in Flash seguendo il paradigma della programmazione Object Oriented ci permette di creare applicazioni scalabili e facili da aggiornare e modificare

Utilizza questo spazio per le tue annotazioni

REQUISITI

Conoscenze richieste

ActionScript 1.0 ed una discreta padronanza nell'utilizzo di Flash Mx e ActionScript 2.0.

Software

Flash Mx 2004 Porfessional

Impegno

Tempo di realizzazione

La programmazione orientata agli oggetti viene indicata spesso e volentieri come un nuovo "paradigma" di programmazione che si distacca notevolmente da quelli utilizzati precedentemente. Se cerchiamo il significato della parola "paradigma" sul nostro vocabolario troveremo riportati tra l'altro termini come esempio, modello, ecc. Per illustrare al meglio i principi fondamentali e i meccanismi che governano la programmazione Object Oriented è il caso di fare un parallelo con la vita reale. Immaginiamo di voler spedire un pacco da Roma a Milano ad un nostro caro amico. Sicuramente ci rivolgeremo ad una ditta di spedizioni limitandoci a consegnare nelle loro mani il nostro pacchetto e il nominativo e l'indirizzo al quale deve essere recapitato senza preoccuparci dei mezzi che la ditta utilizzerà per consegnarlo. Analizzando questo meccanismo e trasponendolo in una forma più consona al paradigma *Object Oriented* abbiamo cercato un "agente" e gli abbiamo passato un "messaggio" contenente una specifica richiesta. Se uno dei compiti di questo agente è quello di soddisfare questo tipo di richieste, al suo interno esisteranno di sicuro uno o più "metodi", ovvero un algoritmo o una serie di azioni da compiere, preposti a risolvere questa particolare esigenza. I dettagli delle azioni che l'agente dovrà compiere per assolvere al compito per cui è preposto assolutamente non ci interessano e questo viene definita come incapsulamento dell'informazione. Un programma sviluppato secondo il paradigma della programmazione orientata agli oggetti può essere visto come comunità di agenti, chiamati oggetti, che interagiscono tra loro scambiandosi messaggi ed eseguendo determinate azioni chiamate metodi. Tutti gli oggetti che compongono il programma sono istanze di una classe e i metodi invocati variano in base ai messaggi ricevuti da un'altra istanza. I concetti fondamentali della programmazione orientata agli oggetti devono essere considerati come degli strumenti per progettare correttamente le nostre applicazioni sviluppate in Flash. Un solo fotogramma è un'applicazione.

Progettare un software Object Oriented significa organizzarlo interamente dividendo le singole funzionalità in classi separate in grado di comunicare tra loro gestendo eventi, variazioni di dati, ecc. Lo strumento che viene utilizzato maggiormente in Flash per organizzare la struttura di un filmato è la timeline che contiene uno o più fotogrammi. Questi, nelle applicazioni più semplici, possono rappresentare i diversi stati del nostro "programma" ma questo approccio ci allontana enormemente dalla programmazione ad oggetti avvicinandoci invece al paradigma utilizzato nella programmazione procedurale visto che determinate azioni verranno eseguite solo quando la testina di riproduzione raggiunge un determinato fotogramma. Come abbiamo più volte ribadito in una applicazione esistono più classi di oggetti, una applicazione Flash sviluppata secondo il paradigma Object Oriented contiene un solo fotogramma che semplicemente carica la classe principale che gestisce tutte le sue funzionalità istanziando altre classi e richiamando metodi e proprietà di queste. Tutte le classi di oggetti che creiamo e utilizziamo in Flash sono dei file *.as* esterni così come le classi aggiunte al software stesso. Seguendo questo percorso sulla nostra macchina *C:\Programmi \Macromedia\Flash MX 2004\en\ First Run\ Classes* scopriremo tutta una serie di file con estensione .as che definiscono le classi non incluse di default nel *Flash Player* come ad esempio *MovieClip, Math, Array,* ecc. All'interno delle nostre applicazioni utilizzeremo vari clip filmato che presumibilmente conterranno alcune animazioni e porzioni di codice inserite nei fotogrammi chiave della loro linea temporale. Abbiamo precedentemente definito questo approccio di programmazione come un'implementazione del paradigma procedurale e per questo potrebbe apparire come una tecnica di sviluppo non corretta. La vera forza di Flash risiede invece proprio

nella sua flessibilità che ci permette di combinare codice progettato seguendo il paradigma orientato agli oggetti e quello procedurale i un ambiente di sviluppo grafico time-based.

DEFINIZIONI

Una classe è modello per la creazioni di oggetti *(istanze)* che hanno in comune tutte le proprietà e i metodi in essa definiti. I metodi sono azioni compiute da un'istanza della classe ed in essa definite. Le proprietà sono variabili definite in una classe di oggetti in cui si memorizzano dati di un determinato tipo. L'ereditarietà è la tecnica utilizzata per organizzare classi di oggetti secondo una precisa gerarchia dove una classe, generalmente indicata come classe figlia, può ereditare tutti i metodi e la proprietà di un'altra detta classe padre. Nella programmazione orientata agli oggetti con i termini interfaccia e implementazione si opera una distinzione tra cosa ci si deve aspettare dall'esecuzione di un metodo e come si è operato per implementarlo.

METODI E PROPRIETÀ: PUBLIC, PRIVATE E STATIC

La nuova versione di ActionScript è quasi del tutto aderente agli standard di programmazione definiti dall'ECMA implementati in quasi tutti i più diffusi e potenti linguaggi Object Oriented. Nella definizione di una classe è ora possibile utilizzare alcune keyword che per i programmatori esperti di *OOP (Object Oriented Programming)* sono di uso comune e considerate essenziali nel processo di sviluppo di una classe di oggetti. Tutti I metodi e le proprietà di una classe sono di default pubblici e quindi richiamabili da tutte le istanze della classe. Proprio per questo non è necessario dichiarare il metodo o la proprietà come public anche se le buone norme di scrittura del codice lo consigliano vivamente. Al contrario le proprietà e i metodi private vanno dichiarati come tali e restano accessibili solo dalla classe stessa e non dalle sue istanze

```
class Cerchio {
  private var raggio:Number;
  function Cerchio (r:Number){
    setRaggio (r); }
  private function setRaggio(val:Number){
    raggio = val; }
  public function getArea():Number{
   var area:Number = Math.PI*(raggio*raggio);
   return area; } }
```

All'interno di questa classe viene dichiarata la proprietà *raggio* come privata ed è pertanto accessibile solo dalla classe stessa, infatti, sempre nella definizione della classe, I due metodi definiti funzionano in maniera differente: Il metodo *setRaggio* viene utilizzato dal costruttore per assegnare un valore alla proprietà raggio della classe stessa e non direttamente accessibile dalle istanze della classe, mentre il metodo *getArea* è accessibile dall'esterno e restituisce il valore dell'area del cerchio. Se in un fla separato creiamo una variabile in cui memorizziamo una *istanza* della classe cerchio e cerchiamo di accedere all'interno di un *trace* ai due metodi otterremo un messaggio di errore nella finestra di output richiamando *setRaggio()* mentre otterremo il valore dell'area accedendo al metodo *getArea()*

```
var test:Cerchio = new Cerchio(8);
// messagio di output : "The member is private and
                              cannot be accessed."
trace(test.setRaggio(8));
// messaggio di output: "201.061929829747"
trace(test.getArea());
```

Il metodo dichiarato come private è accessibile solo dall'interno della classe, ed è questa la principale caratteristica dei membri privati di una classe ovvero quella di essere pensati per un uso interno, il metodo dichiarato come public è invece correttamente utilizzabile dall'esterno e quindi utilizzabile da tutte le istanze della classe. In programmazione esistono però vari work around per evitare errori e aggirare "limitazioni" dei linguaggi e questa regola vale anche per ActionScript 2.0. Se accediamo in maniera dinamica alla proprietà raggio, anch'essa dichiarata come private, non otteniamo nessun errore visto che il controllo viene effettuato in fase di compilazione e non in runtime dal player

```
// nessun messaggio di errore nella finestra di output
trace(test["raggio"]);
```

Un'altra keyword tipica della programmazione orientata agli oggetti è la *keyword static*. Quando una variabile o un metodo vengono dichiarati in questo modo si rendono accessibili direttamente dalla classe e non necessariamente attraverso un'istanza della classe stessa. Alcuni esempi di classi che utilizzano questo tipo di proprietà e metodi sono la classe *Date* e la classe *Math*. Apriamo un nuovo fla e digitiamo nel pannello delle azioni le righe di codice

```
trace(Math.PI);
trace(Date.UTC(2004, 3));
```

Nessuna delle due righe genera errore ed ognuna accede rispettivamente ad una proprietà e ad un metodo dichiarati come static e accessibili senza bisogno di creare un'istanza della classe. Uno dei

TOOLS CONSIGLIATI

SePy
www.sephiroth.it

SciteFlash
http://www.bomberstudios.com/sciteflash/

JEdit
http://www.jedit.org/

Eclipse
www.eclipse.org

XCode
http://blog.pixelconsumption.com/files/Actionscript_Xcode_tutorial.rtf

principali vantaggi derivanti dall'utilizzo dei metodi e delle proprietà static è che in questo modo è possibile creare facilmente dati condivisi tra tutti i membri di una classe. In ActionScript 2.0 è possibile dichiarare il costruttore di una classe come private. In una situazione del genere la classe non espone più un metodo per creare nuove istanze ed è quindi necessario definire un metodo interno che richiami il costruttore. Questa logica nella progettazione di una classe può sembrare paradossale, ma è senza alcun dubbio utile quando si vuole simulare una classe o quando si vuole impostare un limite al numero di istanze che possono essere create dalla classe. Una classe *abstract* è una classe della quale non devono essere create istanze ma che viene utilizzata estendendola in sotto classi

```
class TestAbstract {
  private function TestAbstract(){
    trace("nuova istanza"); }
  public function newInstance(){
    TestAbstract(); } }
```

SUL WEB

www.actionscript.it
www.risorseflash.it
www.ingenium.ws

OVERLOAD DEI COSTRUTTORI E DEI METODI

Nella programmazione Orientata agli Oggetti si indica con il termine *overload* la capacità di definire metodi che implementino diverse azioni a seconda del numero di argomenti che vengono passato o in base al tipo di dati rilevati in un argomento. Nei linguaggi di programmazione più evoluti di ActionScript 2.0, come ad esempio Java, è sufficiente dichiarare più metodi di una classe con lo stesso nome ma con differenti argomenti e relative implementazioni

```
class OverLoader{
  public void test (int x){....}
  public void test (int x, double y){....}
  public void test (string x){....} }
```

Purtroppo AS2 ancora non è così evoluto e l'overload può essere solo simulato. Vediamo con un rapido esempio come sia possibile creare una classe *FourSided.as* che, in base ai parametri passati definisca un quadrato o un rettangolo. Concettualmente quello che andremo a fare è definire due istruzioni condizionali una nel costruttore ed una nel metodo *getArea()* per far si che l'area sia calcolata correttamente e i lati del quadrilatero siano definiti in maniera esatta. L'istruzione condizionale che inseriremo nel metodo *getArea()* e del tutto inutile dal punto di vista della programmazione visto che basta moltiplicare i due lati tra di loro per ottenere l'area del quadrilatero, ma implementeremo ugualmente la simulazione di overload del metodo solo per fini puramente didattici.
Cominciamo ad esaminare la classe partendo dalla sua dichiarazione e dalla definizione delle due proprietà in cui memorizzeremo i valori numerici delle dimensioni dei suoi lati

```
private var _sideA:Number;
private var _sideB:Number;
function FourSided(a:Number, b:Number){} }
```

MUOVIAMO GLI SPRITE

Mettiamo ora in pratica tutte le nozioni apprese definendo una classe che gestisca il movimento di uno sprite all'interno dello stage di un file *.swf* che rimbalzerà quando *"urta" i* bordi del Player.

PREPARIAMO IL FLA

1 **Apriamo Flash e in un nuovo file creiamo un clip filmato al quale assegneremo il linkage palla in modo da poterlo posizionare via codice dalla classe BouncingBall che utilizzeremo per gestire il rimbalzo.**

CREIAMO IL FILE .AS E DEFINIAMO LA CLASSE

```
class BouncingBall
{
  private var _scope:MovieClip;
  private var _clip:String;
  private var _speed:Number;
}
```

2 **Utilizzando il nostro editor preferito o Flash creiamo un nuovo file BouncingBall.as nella stessa cartella che contiene il .fla nel quale definiamo immediatamente la nostra classe e le proprietà in cui memorizzeremo i dati necessari per specificare dove posizionare il clip presente nella libreria *(_scope)*, quale clip filmato utilizzare *(_clip)* e la velocità alla quale deve muoversi *(_speed)*.**

DEFINIAMO GLI ACCESSOR METHODS

```
public function set scope(cl:MovieClip):Void{
  _scope = cl; }
public function get scope():MovieClip{
  return _scope; }
public function set clip(cl:String):Void{
  _clip = cl; }
public function get clip():String{
  return _clip; }
public function set speed(nm:Number ):Void{
  _speed = nm;}
 public function get speed():Number{
  return _speed;
}
```

3 **Abbiamo dichiarato le proprietà della classe come private, definiamo ora dei metodi public per poter accedere ad esse recuperando e/o assegnandogli un valore.**

Prima di affrontare le problematiche inerenti l'overload del costruttore e del metodo, definiamo i due *accessor methods* che ci consentono di modificare i valori delle due proprietà *_sideA* e *_sideB*. Questo tipo di metodi sono utilizzati nella programmazione Object Oriented per mantenere sempre private le proprietà di una classe rendendole accessibili solo attraverso metodi

```
public function get sideA():Number{
    return _sideA; }
public function set sideA(nm:Number):Void{
    _sideA = nm; }
public function get sideB():Number{
    return _sideB; }
public function set sideB(nm:Number):Void{
    _sideB = nm;
}
```

Come prima cosa cominciamo a vedere come deve comportarsi il costruttore a seconda nel numero di parametri che gli vengono passati, per il quadrato ne serve uno solo mentre per il rettangolo sarà necessario indicare le dimensioni di entrambi i lati (base x altezza). L'istruzione condizionale si baserà quindi sulle dimensioni dell'array *arguments* che contiene tutti i parametri passati quando si crea una nuova istanza

```
if(arguments.length < 2){
    sideA = sideB = a;
}else{
    sideA = a;
    sideB = b;
}
```

Nel primo caso (il quadrato) i lati assumeranno lo stesso valore definito dal parametro a, nel secondo caso (rettangolo) assegneremo ai due lati due differenti valori espressi dai parametri *a* e *b*. Analizziamo ora il codice inserito nel metodo *getArea()*. Nel caso del quadrato, ovvero quando i due lati sono uguali, il metodo dovrà restituirci l'elevamento al quadrato del suo lato, al contrario, nel caso del rettangolo, il valore che ci deve restituire sarà generato dalla moltiplicazione dei due lati

```
public function getArea():Number{
    if(sideA == sideB){
        return Math.pow(sideA, 2);
    }else{
        return sideA*sideB;
    }
}
```

Come ho già detto in questo caso sarebbe stato sufficiente moltiplicare in entrambi i casi i due lati per ottenere l'area ma, a fini puramente didattici. Torniamo in Flash e creiamo due istanze della classe e richiamiamo il metodo *getArea()* per vedere come si comporta nel caso gli vengano passati uno o due parametri

```
var square  = new FourSided(6);
trace(square.getArea());
var rectangle  = new FourSided(6, 7);
trace(rectangle.getArea());
```

I messaggi riportati nella finestra di output sono decisamente esaustivi.

Giorgio Natili

- **ACTIONSCRIPT THE DEFINITVE GUIDE FOR FLASH MX** *Colin Moock*
- **ACTIONSCRIPT 2.0 ESSENTIALS** *Colin Moock*
- **OBJECT-ORIENTED PROGRAMMING WITH ACTIONSCRIPT 2.0** *Jeff Tapper*
- **AN INTRODUCTION TO OBJECT ORIENTED Programmino** *Timothy Bud*

DEFINIAMO IL COSTRUTTORE

```
function BouncingBall(sc:MovieClip,
                      cp:String, sp:Number)
{
    scope = sc;
    clip = cp;
    speed = sp;
    init();
}
```

4 **Il costruttore di una classe è una funzione alla quale assegniamo lo stesso nome della classe e nella quale definiamo delle operazioni preliminari. In questo caso specifico utilizziamo gli argomenti che passeremo nel .fla alla nuova istanza per assegnare un valore alle tre proprietà che abbiamo definito e richiamiamo il metodo init**

DEFINIAMO I METODI PRIVATE INIT E MOVINGON

```
private function init():Void{
    scope.attachMovie(clip, "clip_mc", 0, {
    onEnterFrame: movingOn, sp:this.speed,
        hg:Stage.height, wd:Stage.width,
                    signX:1, signY:1}); }
private function movingOn():Void{
    _x += this["sp"]*this["signX"];
    _y += this["sp"]*this["signY"];
    if(_y >= this["hg"] - _height || _y < -
                                _height){
        this["signY"]*= -1;}
    if(_x >= this["wd"] - (_width/2) || _x <- (
                            _width/2)){
        this["signX"]*= -1; }
}
```

5 **Il metodo init posizionerà il clip filmato che simulerà il rimbalzo sullo stage e memorizzerà al suo interno sia la funzione richiamata dall'evento onEnterFrame che i valori utilizzati da questa per gestire il rimbalzo**

CREIAMO UNA NUOVA ISTANZA DELLA CLASSE BOUNCINGBALL

6 **Torniamo nel .fla e nel livello actions dichiariamo una variabile in cui memorizziamo una nuova istanza della classe BouncingBall passandogli come argomenti la _root del filmato (this), il linkage del clip filmato presente nella nostra libreria ("ball") e la velocità con la quale deve essere simulato il rimbalzo**

```
var bball:BouncingBall = new BouncingBall(
                          this, "ball", 4);
```

Come eseguire un ciclo di istruzioni

Le strutture iterative

In questo articolo analizzeremo le strutture del linguaggio, che permettono di eseguire più volte una parte di codice: le strutture iterative

Una situazione ricorrente, che si presenta ad ogni programmatore, è quella di dover ripetere diverse volte, una o più istruzioni all'interno di un'applicazione. Naturalmente possiamo scrivere le stesse istruzioni più di una volta, ma in maniera più comoda ed elegante, possiamo scriverle una volta soltanto per poi dire a VB di ripetere quel gruppo d'istruzioni il numero di volte che ci serve.
Le strutture iterative, o cicli, consentono di eseguire più volte una determinata porzione di codice.
VB .NET 2003 mette a disposizione le seguenti strutture iterative

- *Do…Loop*
- *For…Next*
- *For Each…Next*

La struttura *Do…Loop* permette di eseguire un blocco di istruzioni per un numero di volte che non è necessario definire a priori, infatti, il ciclo terminerà nel momento in cui si verifica una condizione di test che risulti True o False. Nella struttura Do…Loop si possono utilizzare le clausole While ed Until. Utilizzando la clausola While il ciclo terminerà, quando la condizione è True, viceversa, utilizzando la clausola Until il ciclo terminerà, quando la condizione è False. La struttura For..Next è invece la struttura migliore da usare nel caso in cui si devono eseguire per un numero fissato di volte una serie di istruzioni.

LA STRUTTURA DO WHILE..LOOP

La struttura Do While…Loop permette di eseguire un blocco di istruzioni fintanto che risulti True una condizione di test, il ciclo terminerà nel momento in cui la condizione diventa False.

La sintassi è la seguente:

```
Do While Condizione
`Blocco di istruzioni
Loop
```

La prima volta che VB .Net incontra un ciclo Do..While controlla se il valore booleano è True o False.

- se il valore della condizione è False, ignora tutte le istruzioni all'interno del ciclo Do-While.
- se il valore della condizione è True, esegue tutte le istruzioni all'interno del ciclo Do-While.

All'interno del ciclo Do-Loop si esegue l'operazione di divisione, si visualizza, nella ListBox, la lista delle operazioni effettuate e si pone Dividendo = Risultato in modo da continuare a dividere in parti più piccole. Quando VB giunge all'istruzione Loop, si sposta all'istruzione Do-While, valuta nuovamente l'espressione, e se l'espressione è ancora pari a True esegue nuovamente il codice all'interno del ciclo. Se l'espressione è False continua con la prima riga dopo Loop uscendo effettivamente dal ciclo. Ad esempio, ponendo dividendo = 18 e divisore =2, saranno compiute cinque iterazioni fino a quando il

Fig. 1: L'applicazione in esecuzione

Utilizza questo spazio per le tue annotazioni

NOTA

Per aggiungere un elemento al controllo ListBox, è stata usata la sintassi standard che si utilizza per aggiungere un oggetto ad una collezione di oggetti, sintassi che analizzeremo in dettaglio nei prossimi articoli. Vi basti sapere che il risultato sarà quello di visualizzare nel ListBox la stringa racchiusa tra parentesi, che fornisce il dettaglio dell'operazione eseguita. Ad esempio, alla prima iterazione, la stringa visualizzata sarà: 18/2=9

risultato, pari a 0.5625, risulta inferiore ad uno. Se la condizione del ciclo non passa mai da True a False, il ciclo continuerà a ripetersi all'infinito (deadlock) bloccando il programma. Nel nostro esempio si può ottenere un ciclo infinito se poniamo divisore pari ad uno e dividendo ad un qualsiasi valore maggiore o uguale ad uno. Per evitare un ciclo senza fine, si devono controllare le situazioni che possono crearlo (ad esempio con una istruzione If..Then..Else, oppure con una Select..Case), assicurandosi che almeno un'istruzione all'interno del ciclo forzi il valore a True o False, secondo i casi, oppure prevedendo l'uscita forzata dal ciclo. Per il nostro esempio, si può scrivere all'interno del ciclo, l'istruzione:

```
If Divisore = 1 Then Exit Do
```

LA STRUTTURA DO UNTIL..LOOP

La struttura Do-Until si comporta all'opposto del ciclo Do-While, infatti, termina nel momento in cui la condizione di test diventa True.
La sintassi è la seguente:

```
Do Until Condizione
'Blocco di istruzioni
Loop
```

Per avere lo stesso risultato dell'esempio precedente si deve invertire la condizione, pertanto il ciclo diventa:

```
Do Until Risultato <= 1
Risultato = Dividendo / Divisore
ListBoxDati.Items.Add(Dividendo & "/" &
                          Divisore & "=" & Risultato)
Dividendo = Risultato
Loop
```

In questo caso il ciclo viene ripetuto fino a quando Risultato diventa un numero inferiore oppure uguale ad uno. Quando VB arriva all'istruzione Loop ritorna alla riga Do Until e verifica la condizione:

- Se la condizione diventa True il controllo passa alla prima riga dopo l'istruzione Loop, terminando, di fatto, il ciclo.
- Se la condizione diventa False il controllo passa alla prima riga dopo l'istruzione Do Until continuando il ciclo.

In ognuna delle due strutture, sono espressioni accettabili qualsiasi espressione che dia un risultato booleano, così come abbiamo visto nel precedente articolo con l'istruzione If..Then..Else

ESEGUIRE UN CICLO ALMENO UNA VOLTA

Nei nostri programmi, si potrebbe presentare la necessità di ripetere determinate istruzioni almeno una volta. In questo caso, le strutture analizzate fino ad ora, non sono adatte, poiché potrebbero non eseguire alcun ciclo. Per risolvere questo problema,

DIVIDI ET IMPERA

Mettiamo subito in azione la prima struttura creando un nuovo progetto Windows Applications dal nome DividiEtImpera. L'applicazione che andremo a realizzare, dovrà suddividere un numero (dividendo) per un altro numero (divisore), fino a quando il risultato dell'operazione sarà inferiore ad uno. La prima fase è quella del disegno dell'interfaccia utente:

1 Selezioniamo la finestra *Form1* e, se non è già visualizzata, apriamo la finestra delle proprietà. Dalla finestra delle proprietà selezioniamo la proprietà Name e cambiamo subito il nome in: *FormDividi*. Selezioniamo la proprietà *Text* e modifichiamo il testo visualizzato nella barra del titolo in: *Dividi et Impera*

2 Selezioniamo per due volte un controllo *TextBox* dalla casella degli strumenti (nella sezione Windows Form) e disegniamo i due controlli sulla form. Per effettuare l'operazione è sufficiente trascinare i controlli dalla toolbox alla form ponendo attenzione a posizionarli nella maniera desiderata. Gli spigoli della textbox possono essere utilizzati per settare la dimensione

3 Selezioniamo il primo Textbox e, dalla finestra delle proprietà, evidenziamo la proprietà *Name* cambiando il nome in: *TextBoxDividendo*. Evidenziamo la proprietà *Text* e cambiamo il testo nel valore che utilizzeremo. Selezioniamo il secondo *TextBox* e variamo la proprietà *Name* in: *TextBoxDivisore* e la proprietà *Text* nel valore desiderato

Vb.Net mette a disposizione due strutture che eseguono sempre almeno un ciclo, prima di verificare se un valore booleano è True o False.
Ad esempio, usando, il ciclo Do..Loop While si può modificare il codice precedente in:

```
Dividendo = CDbl(TextBoxDividendo.Text)
Divisore = CDbl(TextBoxDivisore.Text)
Do
    Risultato = Dividendo / Divisore
    ListBoxDati.Items.Add(Dividendo & "/" & Divisore &
                                        "=" & Risultato)
    Dividendo = Risultato
Loop While Risultato > 1
```

Il codice appena scritto è, in sostanza, equivalente al precedente, l'unica differenza è nell'assenza dell'istruzione di inizializzazione

```
Risultato = Dividendo
```

Se proviamo a togliere l'istruzione di inizializzazione nel primo esempio, le istruzioni all'interno del ciclo non saranno mai eseguite, poiché la prima volta che viene valutata l'espressione, il valore di Risultato è pari a zero, l'espressione ritorna False ed il ciclo non viene eseguito. Nel secondo caso, quando VB arriva all'istruzione Do, non trova nessuna espressione da valutare quindi passa ad eseguire il codice successivo. Soltanto quando si arriva a Loop While viene valutata l'espressione, e poiché risulta True (Risultato = 9) il controllo passa alla riga successiva all'istruzione Do continuando il ciclo. Allo stesso modo è possibile utilizzare la struttura Do..Loop Until.

IL CICLO FOR-NEXT

Nel caso in cui, si presenta la necessità di eseguire un ciclo per un determinato numero di volte (anche se è sempre possibile utilizzare un ciclo Do-While o Do-Until), la soluzione più semplice è l'utilizzo della struttura For..Next. La sintassi del ciclo For-Next è la seguente:

```
For Contatore =ValoreIniziale to ValoreFinale [Step
                                        Incremento]
  'Blocco di istruzioni da eseguire nel ciclo
Next [Contatore]
```

Gli argomenti Contatore, ValoreIniziale, ValoreFinale

Utilizzando un ciclo For-Next, non si deve mai modificare il valore della variabile all'interno del ciclo, neanche per forzare l'uscita dal ciclo poiché in questo caso è molto meglio utilizzare l'istruzione:

```
Exit For
```

4 **Selezioniamo per due volte un controllo *Label* dalla casella degli strumenti e disegniamo i due controlli sulla form in corrispondenza dei due *TextBox* disegnati in precedenza**

6 **Selezioniamo un controllo *ListBox* dalla casella degli strumenti e disegniamolo sulla form. Questo tipo di controllo consente di effettuare selezioni multiple**

8 **Selezioniamo un controllo Button dalla casella degli strumenti e disegniamolo sulla form. Utilizzzeremo questo bottone per fornire un metodo di controllo all'utente**

5 **Selezioniamo la prima Label e, dalla finestra delle proprietà, evidenziamo la proprietà *Text* per cambiare il testo visualizzato in: *Dividendo*. Allo stesso modo selezioniamo la seconda Label e variamo la proprietà *Text* in: *Divisore***

7 **Selezioniamo il controllo *ListBox* e, dalla finestra delle proprietà, variamo la proprietà *Name* in: *ListBoxDati*. Da questo momento in poi tutti i riferimenti nel codice alla listbox possono essere accessibili tramite il nome listboxdati**

9 **Selezioniamo il controllo *Button*, dalla finestra delle proprietà, modifichiamo la proprietà *Name* in: *ButtonDividi* e la proprietà *Text* in: Dividi Et Impera. Utilizzeremo questo controllo per gestire l'evento click che ci restituirà l'esito del tentativo**

e Incremento sono valori numerici. La clausola Step non è obbligatoria, se viene omessa il valore di default di Incremento sarà pari ad uno. È possibile, anche se non è obbligatorio, specificare l'argomento Contatore nell'istruzione Next per migliorare la leggibilità del programma (naturalmente è necessario specificare la stessa variabile presente nell'istruzione For).
Analizziamo il funzionamento della struttura:

- Appena si entra nel ciclo, la variabile Contatore assume il valore specificato in ValoreIniziale
- Viene verificato che il valore della variabile Contatore sia minore o uguale del valore della variabile ValoreFinale (in caso contrario le istruzioni presenti nel ciclo non vengono eseguite e l'esecuzione passa immediatamente all'istruzione successiva all'istruzione Next)
- Vengono eseguite le istruzioni all'interno della struttura.
- Viene incrementato il Contatore del valore specificato in Incremento.

Queste operazioni sono eseguite fino a quando il valore di Contatore supera il valore di ValoreFinale Il valore di Incremento può essere anche negativo, in tal caso il ciclo termina solo quando il valore della variabile Contatore è strettamente minore del ValoreFinale. Per mettere alla prova la struttura For..Next, aggiungiamo alla nostra applicazione di esempio un TextBox (che chiameremo TextBoxNumeroVolte) ed un controllo Button (che chiameremo ButtonNumeroVolte). In questo caso, l'utente dovrà scrivere in TextBoxNumeroVolte il numero di volte che intende compiere l'operazione di divisione. Come di consueto, scriviamo il codice all'interno della procedura di evento *ButtonNumeroVolte_Click*

```
Dim Risultato As Double
Dim Dividendo As Double
Dim Divisore As Double
Dim Indice As Integer
Dividendo = CDbl(TextBoxDividendo.Text)
Divisore = CDbl(TextBoxDivisore.Text)
For indice = 1 To CInt(TextBoxNumeroVolte.Text)
    Risultato = Dividendo / Divisore
    ListBoxDati.Items.Add(Dividendo & "/" &
                          Divisore & "=" & Risultato)
    Dividendo = Risultato
Next
```

Anche in questo caso le parti salienti del codice sono rimaste uguali l'unica differenza è l'introduzione della struttura For..Next (oltre alla dichiarazione della variabile Indice di tipo Integer)

- Si avvia il ciclo con la parola chiave For, utilizzando la variabile Indice come contatore e fissando gli estremi da uno al valore immesso dall'utente in TextBoxNumeroVolte (supponiamo ad esempio quattro). La prima volta che si entra nel ciclo la variabile indice assumerà, quindi, il valore uno.
- Le istruzioni all'interno del ciclo, sono sempre le stesse dell'esempio precedente, che si occupano di eseguire i calcoli e visualizzare nella ListBox, la lista delle operazioni effettuate
- L'ultima riga contiene la parola chiave Next. Quando VB incontra l'istruzione Next, incrementa il valore della variabile Indice, portandolo in questo caso a due e controlla se tale valore è inferiore o uguale all'estremo superiore (quattro) in caso affermativo il codice riparte dalla riga successiva a quella contenente l'istruzione For, in caso contrario il controllo passa alla riga successiva all'istruzione Next uscendo dal ciclo.

Per uscire dal ciclo prima della sua fine naturale si può usare l'istruzione Exit For, in questo caso il controllo dell'applicazione passa all'istruzione immediatamente successiva all'istruzione Next. Il funzionamento del ciclo For Each…Next è simile a quello del ciclo For…Next ma invece di ripetere le istruzioni un numero fissato di volte, ripete le istruzioni per ciascun elemento di un insieme o di una matrice come vedremo meglio in uno dei prossimi articoli.

Fig. 2: *L'applicazione con il ciclo For..Next in esecuzione*

CICLI NIDIFICATI

È probabile che in alcuni casi, si renda necessario inserire un ciclo in un altro ciclo. Ogni volta che racchiudiamo un ciclo in un altro, si parla di cicli nidificati ed il funzionamento è simile alla nidificazione dell'istruzione If..Then..Else vista nell'articolo del mese scorso. Nei cicli nidificati, si conclude prima il ciclo più interno e poi a cascata tutti gli altri.

NOTA

L'operatore di concatenazione & permette di concatenare più espressioni restituendo un risultato di tipo String. Il risultato di una operazione di concatenazione è una stringa formata dalla concatenazione dei due operandi da sinistra a destra. Ad esempio l'espressione:

```
Messaggio = "Elemento n° " & 4
```

Restituisce la stringa: *Elemento n° 4*

Vb .Net mette a disposizione un'ulteriore struttura iterativa, la struttura While..End While L'istruzione While (analogamente a Do While..Loop)consente di eseguire un blocco di istruzioni per un numero indefinito di volte, in base al valore booleano di una condizione. Le istruzioni sono ripetute fino a quando la condizione rimane True.
La sintassi è:
While condizione
[istruzioni]
End While
Per uscire dall'istruzione While si può utilizzare l'istruzione Exit While.

Ad esempio:

```
Dim Indice1 As Integer
Dim Indice2 As Integer
For Indice1 = 1 To 3
    For Indice2 = 1 To 2
        MessageBox.Show(Indice1 & "-" & Indice2)
    Next
Next
```

Visualizza i messaggi: *1-1, 1-2, 2-1, 2-2, 3-1, 3-2* che dimostra, appunto, come il ciclo più esterno incrementa il proprio contatore soltanto dopo che il ciclo più interno è terminato. È possibile nidificare più di due cicli *For* ed è inoltre possibile combinarli con cicli *Do..While* in modo da avere dei cicli For nidificati in un ciclo *Do..While* e viceversa. Il consiglio è sempre quello di non approfittarne per scrivere codice illeggibile. È possibile nidificare più di due cicli *For* ed è inoltre possibile combinarli con cicli *Do..While* in modo da avere dei cicli *For* nidificati in un ciclo *Do..While* e viceversa. VB non pone particolari limiti, e ci permette di raggruppare tutti i cicli che vogliamo, il consiglio è sempre quello di non approfittarne per scrivere codice illeggibile e di stare attenti ad applicare dei rientri a ciascun ciclo. In questo modo si può individuare più facilmente dove comincia e dove finisce ogni ciclo. Ad esempio osserviamo l'aspetto caotico dei cicli nidificati senza rientri:

```
Do While Rivista = "Io Programmo"
Do
For i = 1 To NumeroPagine
SfoglioLaRivista
Next
Loop Until Leggo = "No"
Loop
```

Scrivendo lo stesso codice con i rientri otteniamo:

```
Do While Rivista = "Io Programmo"
  Do
    For i = 1 To NumeroPagine
      SfoglioLaRivista
    Next
  Loop Until Leggo = "No"
Loop
```

Dal punto di vista di VB, entrambi gli esempi funzionano allo stesso modo, ma dal punto di vista del programmatore il secondo esempio è molto più semplice da leggere e da capire.

CONCLUSIONI

In questo articolo abbiamo analizzato le strutture che permettono di ripetere l'esecuzione di un gruppo di istruzioni. Nel prossimo articolo termineremo la sintassi di VB.NET introducendo l'uso di procedure e funzioni.

Luigi Buono

FINESTRA DEL CODICE

```
Private Sub ButtonDividi_Click(ByVal sender
        As System.Object, ByVal e As
        System.EventArgs) Handles
        ButtonDividi.Click
End Sub
```

1 **Per scrivere il codice è sufficiente fare doppio click sul controllo Button. Con questa operazione si aprirà la finestra del codice con il cursore posto all'interno della procedura di evento ButtonDividi_Click**

DICHIARAZIONE DELLE VARIABILI

```
Dim Risultato As Double
Dim Dividendo As Double
Dim Divisore As Double
```

2 **Definiamo le variabili Risultato, Dividendo e Divisore di tipo Double.**

INIZIALIZZAZIONE DELLE VARIABILI

```
Dividendo = CDbl(TextBoxDividendo.Text)
Divisore = CDbl(TextBoxDivisore.Text)
Risultato = Dividendo
```

3 **Poniamo le variabili Dividendo e Divisore pari al valore inserito nel corrispondente TextBox. Poiché i valori immessi in un TextBox sono di tipo String, dobbiamo utilizzare la funzione di conversione CDbl che converte una qualsiasi espressione in un valore di tipo Double. La variabile Risultato viene inizializzata al valore del Dividendo per il motivo che vedremo in seguito.**

DICHIARAZIONE DELLE VARIABILI

```
Do While Risultato > 1
    Risultato = Dividendo / Divisore
    ListBoxDati.Items.Add(Dividendo &
        "/" & Divisore & "=" & Risultato)
    Dividendo = Risultato
Loop
```

4 **Definiamo le variabili Risultato, Dividendo e Divisore di tipo Double. L'istruzione Do While valuta la condizione booleana: Risultato > 1 e, fin tanto che la condizione è vera, vengono eseguite le istruzioni all'interno del blocco. Quando la condizione è falsa (in questo caso, quando il risultato è minore o uguale ad uno) il controllo salta all'istruzione successiva alla parola chiave Loop.**

NOTA

Nella struttura *For..Next* le espressioni di *ValoreIniziale*, *ValoreFinale* e *Step* sono valutate una sola volta, alla prima occorrenza dell'istruzione *For*, e non saranno più valutate, neppure se nel ciclo sono presenti istruzioni che ne modificano il valore.

Quando il numero di cicli nidificati diventa elevato, si deve porre particolare attenzione affinchè i cicli interni non modifichino inavvertitamente le condizioni o le variabili di calcolo dei cicli più esterni. In caso contrario, si potrebbe generare un comportamento anomalo ed essere costretti ad esaminare tutti i cicli.

Impara a manipolare i file con Java

Leggere e scrivere i file

Quasi tutti i programmi del mondo reale leggono e scrivono dati sul disco rigido. In alcuni linguaggi è facilissimo lavorare con i file. In Java lo è un po' meno

Utilizza questo spazio per le tue annotazioni

Conoscenze richieste

Minima conoscenza dell'ereditarietà, polimorfismo

Software

Java 2 Standard Edition SDK 1.4 o superiore

Impegno

Tempo di realizzazione

Uno tra gli scogli che attendono i programmatori principianti è la gestione dei file. In Java non basta chiamare un paio di funzioni per aprire un file e per leggerne il contenuto. Prima di accedere ai dischi si devono capire alcune cose. Che non sono banali, ma - per fortuna – nemmeno troppo difficili. Le classi di cui parleremo sono nel package *java.io*, che si occupa dell'Input/Output e può incutere timore ai turisti delle librerie di Java. Possibile che servano tante classi per delle banali operazioni sul file system? Il package java.io è complesso per due motivi. Primo: in Java tutti i metodi di Input/Output si somigliano. Java ha un meccanismo generico per leggere e scrivere, e *java.io* include molte varianti di questo meccanismo. Questa è una buona cosa, perché ci permette di scrivere codice che legge nello stesso modo da un file o da un sito web. Il secondo motivo è che un programma Java deve poter leggere e scrivere i file nello stesso modo su qualsiasi piattaforma. I file system sono particolarmente legati ai sistemi operativi, quindi i progettisti della libreria hanno fatto un po' di fatica per ottenere un sistema che funzionasse dignitosamente su qualsiasi sistema operativo. Il risultato non poteva essere troppo semplice.

FILE A CHI?

Il package *java.io* contiene una classe ingannevole: *File*. No, non rappresenta un file – tanto è vero che non contiene metodi come *open()* o *read()*. *File* rappresenta invece il descrittore di un file o di una directory. I descrittori contengono "meta-informazioni" sui file: se il file è protetto in scrittura, se si tratta di un file vero e proprio o di una directory, e così via.

```
import java.io.File;
public class DemoDirectory {
    public static void main(String[] args) {
        assert args.length == 1 : "Numero di argomenti
                                                errato";
        String nome = args[0];
        File dir = new File(nome);
        assert dir.exists() : nome + " non esiste";
        assert dir.isDirectory() : nome + " non è una
                                            directory";
        assert dir.canRead() : "Non puoi leggere da
                                             " + nome;
        stampaInformazioni(dir);
        elencaContenuto(dir);
        creaSottodirectory(dir);
        System.out.println("OK");
    }
    private static void stampaInformazioni(File dir) {
        System.out.println("Directory " +
                                  dir.getAbsolutePath());
        System.out.println("Ultima modifica:
                                  " + dir.lastModified());
    }
    private static void elencaContenuto(File dir) {
        String[] contenuto = dir.list();
        for (int i = 0; i < contenuto.length; i++) {
            System.out.println(contenuto[i]);
        }
    }
    private static void creaSottodirectory(File dir) {
        assert dir.canWrite() : "Non puoi scrivere in " + dir;
        File sottodirectory = new File(dir, nomeCasuale());
        sottodirectory.mkdir();
    }
    private static String nomeCasuale() {
        return "x" + (int)(Math.random() * 10000);
    }
}
```

Il programma vuole che gli si passi il percorso di una directory. Lo puoi lanciare così:

```
java -ea Directory c:/test
```

Il parametro *-ea* sulla riga di comando abilita le asserzioni (vedi il box sull'argomento). *"DemoDirectory"* è il nome del programma. Segue l'unico argomento passato al programma: il percorso della directory. Se usi Windows puoi usare sia *"c:/test"* che *"c:\test"*. Il programma cerca un parametro sulla riga di comando. Se lo trova usa la stringa per creare un oggetto *File*. A questo punto non è detto che il *File* rappresenti una directory, che questa directory esista, o che sia leggibile. Il programma controlla tutte queste eventualità con delle asserzioni. Se una qualsiasi asserzione fallisce, il programma si ferma con un errore. Il metodo *stampaInformazioni()* stampa il percorso assoluto e la data di ultima modifica della directory. Il metodo *elencaContenuto()* elenca il contenuto della directory (come il comando DOS *dir*) sotto forma di un array di nomi di file e sottodirectory. Il tutto finisce stampato sullo schermo. Infine, il metodo *creaSottodirectory()* verifica che la directory sia scrivibile e crea una sottodirectory. Per farlo costruisce una nuova istanza della classe *File()*. Il costruttore che abbiamo usato prende la directory madre e il nome della directory *figlia* (*"x"* seguita da un numero casuale). A questo punto la directory esiste nel programma, ma non sul disco rigido. Per crearla si usa il metodo *File.makeDir()*. Prova a far girare *DemoDirectory* due o tre volte e vedrai nascere delle sottodirectory con brutti nomi sotto *c:/test*. Quindi la classe *File* si occupa delle directory e delle operazioni sui file, non sul loro contenuto. Per scrivere o leggere un file si deve usare uno strumento diverso: gli *stream*.

IL MONDO È FATTO A TUBI

Uno stream è un tubo che trasporta dati. Un capo dello stream è collegato ad una sorgente, l'altro è collegato ad una destinazione. A seconda di quale capo del tubo abbiamo in mano, possiamo scrivere o leggere i dati. Non sappiamo come questi dati verranno trasportati o elaborati, ma quello che importa è che dall'altra parte c'è qualcuno che vuole riceverli (se siamo una sorgente) o spedirli (se siamo una destinazione). Quando dobbiamo comunicare apriamo uno stream, lo usiamo e poi lo chiudiamo. In Java esistono due categorie di stream: quelli che ragionano "a byte" e quelli che ragionano "a caratteri" (le due cose non coincidono, perché un carattere Java è un intero).
Gli stream del primo tipo sono figli delle interfacce *InputStream* e *OutputStream;* quelli del secondo sono figli delle interfacce *Reader* e *Writer.*

```
import java.io.FileWriter;
import java.io.IOException;
import java.io.Writer;
public class PiccoloFile {
    public static void main(String[] args) throws
                                        IOException {
        Writer out = new FileWriter("piccoloFile.txt");
        out.write('t');
        out.write('e');
        out.write('s');
        out.write('t');
        out.close(); }
}
```

Questo programma crea un *FileWriter,* uno stream di output basato sui caratteri (è un *Writer*) e collegato ad un file. Ora il *main()* è la sorgente e il file è la destinazione. Possiamo anche "dimenticare" con chi stiamo parlando: il programma fa un upcast dello stream all'interfaccia *Writer,* che è la stessa per uno stream basato su un file o per uno basato su una connessione Internet. *PiccoloFile* scrive dei char sullo stream con il metodo *Writer.write()*. In realtà il metodo prende degli int, ma Java converte automaticamente i char nei loro codici interi. Prova ad aprire *piccoloFile.txt* dopo aver fatto girare il programma. Prima di terminare, il programma chiama *Writer.close()* per chiudere lo stream. È importante ricordare di farlo: a nessuno fa piacere un file che resta bloccato in lettura, o una connessione di rete che rimane aperta.

UNA QUESTIONE DI PRESTAZIONI

Purtroppo le operazioni di I/O possono diventare molto lente:

```
import java.io.FileWriter;
import java.io.IOException;
import java.io.Writer;
public class GrandeFile {
    private static final int DIMENSIONE = 100000000;
    public static void main(String[] args) throws
                                        IOException {
        long tempoIniziale = System.currentTimeMillis();
        Writer out = new FileWriter("grandeFile.txt");
        for(int i = 0; i < DIMENSIONE; i++)
            out.write('x');
        out.close();
        long tempoFinale = System.currentTimeMillis();
        System.out.println("Tempo: " + (
                (tempoFinale - tempoIniziale) / 1000.0)); }
}
```

Questo programma riempie un file con cento milioni di caratteri *'x'*, e usa il metodo statico *System.currentTimeMillis()* per misurare il tempo richiesto dall'operazione. *System.currentTimeMillis()* re-

- **Imparerai ad usare la classe File.**
- **Farai conoscenza con gli stream.**
- **Scoprirai i decoratori**

I/O ED ECCEZIONI

Un sacco di cose possono andare storte mentre si lavora con gli stream: il disco rigido può riempirsi, la connessione di rete può cadere, eccetera. Per questo motivo, quando si lavora con l'I/O si deve sempre fare attenzione alle eccezioni. Le eccezioni degli esempi di questo mese sono gestite in modo decisamente rozzo: buttandole fuori dal main(). Le eccezioni checked che possono saltare fuori da questi programmi sono tutte sottoclassi di IOException, quindi basta usare throws IOException per gestirle tutte in un colpo solo. Questo approccio sarebbe inaccettabile in un programma vero. Le operazioni di I/O sono la classica situazione nella quale si devono gestire attentamente tutti gli errori. Se la scrittura di un file fallisce, ad esempio, potremmo chiedere all'utente se vuole annullare o riprovare l'operazione. Purtroppo questo genere di cose sono le prime che vengono dimenticate quando c'è poco tempo per scrivere il codice, e il risultato sono programmi fragili e utenti frustrati.

ESERCIZIO 1

Per rendere tutto il più semplice possibile, abbiamo lanciato DemoDirectory con il percorso assoluto della directory c:/test. È possibile anche usare un un percorso relativo:

```
java -ea Directory test
```

Se si usa un percorso relativo, dove devi mettere la directory test perché il programma la veda?

ESERCIZIO 2

Fai girare *GrandeFile* e *GrandeFileVeloce* sulla tua macchina. I tuoi risultati sono simili ai miei? Quanti rpm ha il tuo disco rigido?

stituisce il tempo attuale misurato in millisecondi a partire da una data convenzionale. L'ultima riga del programma sottrae il tempo finale a quello iniziale, e lo divide per 1000 per ottenere il numero di secondi trascorsi. Abbiamo specificato che 1000 è un numero decimale per evitare l'arrotondamento di una divisione intera: ci interessano anche i decimi di secondo. Ho provato il programma sulla mia macchina. È un portatile, ma ha un disco rigido a 7200 rpm. Il programma impiega più di 35 secondi per scrivere un file di 100 Mb. Il problema è che stiamo scrivendo un carattere alla volta. Possiamo risparmiare molto tempo e alleggerire la vita del nostro povero disco rigido usando un buffer, un'area di memoria che conserva i caratteri che scriviamo e li scrive periodicamente "in un colpo solo". Tra le classi di *java.io* ci sono anche un *Reader* e un *Writer* bufferizzati: *BufferedReader* e *BufferedWriter.* Useremo il primo. BufferedWriter è una bestia particolare. È un *Writer,* ma il suo costruttore prende un altro *Writer. BufferedWriter* è quindi un *Writer* che contiene un *Writer.* Quando facciamo una scrittura, il *BufferedWriter* conserva i dati nel suo buffer. Quando il buffer si riempie, *BufferedWriter* scarica tutti i dati sul Writer al quale si appoggia. In pratica questo oggetto si "avvolge" intorno ad un altro *Writer* e gli aggiunge delle funzionalità nuove. Questo è un design classico, che si chiama di solito *wrapper,* o *decorator:* l'oggetto esterno "decora" quello interno aggiungendogli alcune funzionalità. Se non conosci il pattern decorator, leggi il box che ne parla in questo articolo. Ecco una versione di *GrandeFile* che usa un *BufferedReader:*

```
import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;
import java.io.Writer;
public class GrandeFileVeloce {
  private static final int DIMENSIONE = 100000000;
  public static void main(String[] args) throws
                                        IOException {
    long tempoIniziale = System.currentTimeMillis();
    Writer out = new BufferedWriter(
                    new FileWriter("grandeFile.txt"));
    for(int i = 0; i < DIMENSIONE; i++)
      out.write('x');
    out.close();
    long tempoFinale = System.currentTimeMillis();
    System.out.println("Tempo: " + (
        (tempoFinale - tempoIniziale) / 1000.0)); }
}
```

Sul mio PC, questa variante del programma richiede circa 10 secondi, che è meno di un terzo del tempo precedente. Ho provato a fare un esperimento simile usando i decoratori con buffer per gli stream che "ragionano in byte": *BufferedOutputStream* e *BufferedInputStream.* I risultati in questo caso sono ancora migliori. Quando si usa uno stream "bufferizzato" è particolarmente importante ricordare di chiuderlo alla fine. Se non lo facessimo, parte dei dati potrebbero restare nel buffer, e il risultato sarebbe un file incompleto. Quando chiudiamo lo stream, Java si occupa di fare il "flush" del buffer, cioé di trasferirne tutto il contenuto all'altro capo dello stream.

ALTRE DECORAZIONI

Java offre diversi decoratori che prendono altri stream e gli aggiungono funzionalità. Alcuni ten-

ASSERZIONI

Negli esempi di questo articolo abbiamo usato una nuova utilissima parola chiave: assert. È al tempo stesso una forma di documentazione e un metodo per fare il debugging. Si usa così:

```
assert <espressione booleana> : <messaggio>
```

che significa: mi aspetto che l'espressione booleana sia vera. Se questo non succede, si tratta di un bug: segnalamelo con questo messaggio di errore. Ad esempio, una classe ContoCorrente potrebbe avere un metodo come questo:

```
public void emettiAssegno(double importo) {
  assert valuta() >= importo : "Assegno scoperto".
  (operazioni sul conto)
}
```

emettiAssegno() si aspetta che l'importo dell'assegno non sia maggiore della valuta sul conto. Questa è documentazione, perché dice a chi legge: "questo metodo non gestire questo errore, quindi chi chiama il metodo deve essere sicuro che l'assegno è coperto"; ma è anche codice, perché se il chiamante non verifica che l'assegno sia coperto, il programma genere un errore con un messaggio che lo spiega chiaramente. Non è un'eccezione, è proprio un errore: per l'esattezza, un AssertionError. Le asserzioni non servono a gestire circostanze strane, ma veri e propri bug. Le asserzioni sono utili quando testiamo il programma. Quando lo rilasciamo, ci aspettiamo (o illudiamo) che non ci siano errori e che tutte le asserzioni siano vere. Non vogliamo che il nostro codice venga rallentato dalle asserzioni. Per questo motivo, le asserzioni sono normalmente disabilitate. Possiamo abilitarle durante lo sviluppo lanciando la virtual machine con il parametro -enableassertions, o più brevemente -ea.
Quando la Virtual Machine non trova questo parametro, tutte le asserzioni vengono ignorate. Durante lo sviluppo conviene sempre mantenere "accesi" gli assert.

gono il conto dei numeri di riga *(LineNumberReader)*, altri comprimono o decomprimono i dati in formato ZIP *(java.util.zip.ZipInputStream)*, altri ancora leggono e scrivono interi oggetti Java con un meccanismo chiamato serializzazione *(ObjectInputStream)*. Esistono anche classi come *OutputStreamWriter* che convertono uno stream dal mondo dei byte a quello dei caratteri. Non è raro vedere più livelli di decorazione intorno ad un solo stream.

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.LineNumberReader;
import java.io.StringReader;
public class Dante {
    public static void main(String[] args) throws
                                              IOException {
        String testo = "Ahi serva Italia, di dolore ostello";
        LineNumberReader in = new LineNumberReader(
                new BufferedReader(
                        new StringReader(testo)));
        saltaCaratteri(in, 10);
        stampa(in);
    }
    private static void saltaCaratteri(LineNumberReader in,
                              int num) throws IOException {
        in.skip(num); }
    private static void stampa(Reader in) throws
                                              IOException {
        int c = in.read();
        while(c != -1) {
            System.out.print((char)c);
            c = in.read();
        }
        in.close();
        System.out.println(); // vai a capo
```

```
    }
}
```

Questo programma legge da un *LineNumberReader*, che decora un *BufferedReader*, che a sua volta decora uno *StringReader*. Lo *StringReader* è un *Reader* basato su una stringa. Non sembra molto utile, ma il vantaggio diventa evidente quando si pensa che il metodo *stampa()* funziona nello stesso modo sia che i dati vengano da una stringa che da un file: legge un carattere alla volta finché non riceve un carattere -1, che segnala la fine dello stream. Il *BufferedReader*, invece, è davvero inutile, perché la stringa è già in memoria.
L'ho usato solo per dimostrare che si possono avvolgere più decoratori intorno allo stesso oggetto. Il decoratore più esterno è un *LineNumberReader*, che permette tra l'altro di saltare dei caratteri con il metodo *skip()*. Il programma elimina i primi 10 caratteri e stampa il contenuto dello stream. A differenza di *read()*, il metodo *skip()* non appartiene all'interfaccia *Reader:* è una specialità di *LineNumberReader*. Il metodo *saltaCaratteri()*, che usa *LineNumberReader.skip()*, deve sapere di avere a che fare con un *LineNumberReader*.
Il metodo *stampa()*, che usa *Reader.read()*, funziona invece con qualsiasi *Reader*.

CONCLUSIONI

Con questo articolo si conclude il nostro lungo corso introduttivo a Java. A partire dal mese venturo tratteremo argomenti più avanzati. Ci aspetta una mini-serie sui thread e la programmazione parallela. Non mancate!

Paolo Perrotta

Scrivi due programmi simili a *GrandeFile* e *GrandeFileVeloce* usando la gerarchia *InputStream/ OutputStream*. Quali sono i risultati?

Scrivi una versione del programma Dante che legge il testo da un file anziché da una stringa. Quante righe di codice devi cambiare?

IL PATTERN DECORATOR

L'idea di "decorare" un oggetto con un altro oggetto che ha la stessa interfaccia è un classico della programmazione object-oriented. In altre parole, è quello che si dice un pattern. Ecco un diagramma UML del pattern Decorator: Decoratore eredita da Componente, che di solito è una classe astratta o un'interfaccia. Decoratore conserva l'istanza di un ComponenteConcreto (l'oggetto decorato) che gli viene passato in fase di costruzione. Un metodo operazione() sul Decoratore chiama di solito l'operazione() del Decorato, ma aggiunge alcune funzionalità specifiche. Un ComponenteConcreto può essere a sua volta un Decoratore, quindi le funzionalità possono essere aggiunte "a strati". Dal punto di vista del client, questa è una bella cosa per due motivi. Primo: l'oggetto acquista nuove proprietà (ad esempio un InputStream acquista un buffer). Secondo, lo fa in modo trasparente. Il Decoratore è un Componente proprio come il ComponenteConcreto, quindi si può usare in tutti i punti del codice che parlano ai Componenti. In altre parole, questo pattern aggiunge funzionalità all'oggetto senza toccare il codice che lo usa. Un Decoratore può anche aggiungere metodi all'interfaccia Componente, come avviene per il LineNumberReader di cui parliamo nell'articolo. In questo caso, però, si perde in parte la sostituibilità: chi vuole usare i nuovi metodi deve sapere di avere a che fare con uno specifico Decoratore.

I trucchi del mestiere

Tips & Tricks

Questa rubrica raccoglie trucchi e piccoli pezzi di codice, frutto dell'esperienza di chi programma, che solitamente non trovano posto nei manuali. Alcuni di essi sono proposti dalla redazione, altri provengono da una ricerca su Internet, altri ancora ci giungono dai lettori. Chi volesse contribuire, potrà inviare i suoi Tips&Tricks preferiti. Una volta selezionati, saranno pubblicati nella rubrica. Il codice completo dei tips è presente nel CD allegato nella directory *tips*\\ o sul Web all'indirizzo: cdrom.ioprogrammo.it.

PILOTARE UNA PAGINA DI LOGIN DA VISUAL BASIC

Di seguito riporto il codice di esempio da cui poter prendere spunto per pilotare da Visual Basic una pagina web. Nel caso sotto riportato, utilizzando il componente *Microsoft Internet Controls (shdocvw.dll)* e il riferimento a *Microsoft HTML Object Library (mshtml.tlb)*, visualizzo una pagina di login (recuperando l'url da linea di comando), riempio le caselle di testo *'uetnte'* e *'password'*, e forzo il click sul pulsante di connessione.
In questo modo l'applicativo si logga automaticamente al sito, navigandolo all'interno del controllo *WebBrowser* a tutto schermo.
In fase di progettazione, l'unica cosa da aggiungere al form è il controllo *WebBrowser;* per mia comodità operativa, nelle proprietà del Form ho impostato l'esecuzione a tutto schermo.

Tip fornito dal sig. Diego Cozzi

```
Option Explicit

Private defaultURL As String
Private Sub Form_Activate()
    'defaultURL = "http://srvweb/intranet/default.asp"
    defaultURL = Command

    DimensionaWebBrowser

    Me.WebBrowser1.Navigate (defaultURL)
End Sub

Private Sub DimensionaWebBrowser()
    Me.WebBrowser1.Top = 10
    Me.WebBrowser1.Left = 10
    Me.WebBrowser1.Height = Me.Height
    Me.WebBrowser1.Width = Me.Width
End Sub

Private Sub AutoConnessione()
    Dim HTMLDoc As HTMLDocument

    On Error GoTo ErrConnessione
```

IL TIP DEL MESE

VALIDAZIONE DI UNA FORM

Molte volte un programmatore ha l'esigenza di controllare che tutti i campi di un modulo Web siano stati compilati o meno, questo tipo di problema poteva essere risolti scrivendo uno script javascript (con tanti IF)… ma con questo file JS tutto questo è terminato! Basta inserire un semplice tag (esempio required=yes) per rendere obbligatorio il campo senza dover scrivere piu' una riga di codice!

Tip fornito dal sig. Rosario Sensale
Codice sul CD: *validatorform.zip*
File Contenuti nello Zip: *FormValidator.js* (libreria per il controllo) *Language.js* (msg), *TestForm.htm* (demo)

Lo script funziona con: caselle di testo, aree di testo, caselle a discesa, ma puo' essere implementato anche per tutti gli altri oggetti. Lo script permette di controllare (per ogni oggetto):

- il campo deve essere compilato;
- il campo deve contenere almeno x caratteri;
- il campo deve contenere al max y caratteri;
- il campo deve contenere un indirizzo mail;

Compatibilità: lo script è stato provato con IE 6, Mozilla FireFox, Netscape 7.2

```
   Set HTMLDoc = Me.WebBrowser1.Document

   If HTMLDoc Is Nothing Then
      MsgBox "Impossibile Ottenete l'oggetto Document"
      GoTo ErrConnessione
   Else
      HTMLDoc.All("utente").Value = "usrCozzi"
      HTMLDoc.All("password").Value = "pwdCozzi"

      HTMLDoc.All("entra").Click

      defaultURL = "" 'lo resetto così non rischio di non potermi
            collegare con un altro utente se dovessi ritornare sulla
                                                  pagina di default

      Set HTMLDoc = Nothing
   End If

   Exit Sub
ErrConnessione:
   MsgBox Err.Description
End Sub

Private Sub WebBrowser1_DocumentComplete(ByVal pDisp As
                                    Object, URL As Variant)
   If defaultURL = URL Then
      AutoConnessione
   End If
End Sub
```

GESTIRE UTENTI E GRUPPI IN UNA RETE

Questo progetto Visual Basic permette di gestire gli utenti presenti in un dominio Windows 2k Server.
Elenca gli utenti e i gruppi e permette di effettuare alcune operazioni su questi (esempio blocco/sblocco di un account). Il file ZIP contiene tutte le librerie necessarie.

Tip fornito dal sig. Rosario Sensale

Codice allegato: *WinUser IoProgrammo.zip*

PHP

COME MANDARE UNA MAIL IN PHP

Quanto segue vale solo per coloro che utilizzano PHP 4 sotto windows: andate a cercare il file php.ini e modificate le seguenti righe.

Tip fornito dal sig. Dario Fadda

```
[mail function]
SMTP           =    ;for win32 only
sendmail_from    =    ;for win32 only
```

In corrispondenza della voce SMTP dovrete scrivere localhost (oppure il nome del server di posta in uscita). In corrispondenza della voce sendmail_from invece scrivete il vostro indirizzo di posta. Così permetterete che la funzione di PHP *mail()* funzioni a dovere. Ora, dopo questa semplice configurazione, iniziamo ad entrare nel vivo della guida. Seguite i passi seguenti per creare, prima un semplice modulo di feedback in HTML (che ci servirà come interfaccia grafica per il nostro script), e poi lo script vero e proprio.

Un semplice modulo di feedback

Con un editor di testo digitiamo:

```
<html>
  <head><title>Semplice modulo di feedback</title>
  </head>
    <body>
      <form method="post" action="invia_modulo.php">
        <p><b>Il tuo nome:<b><br>
        <input type ="text" name="nome_mittente" size=30></p>
        <p><b>Il tuo indirizzo e-mail:</b><br>
        <input type="text" name="mail_mittente" size=30></p>
        <p><b>Messaggio:</b><br>
        <textarea name="messaggio" cols=30 rows=50
                                  wrap=virtual></textarea></p>
        <p><input type="submit" name="invio" value="ok"></p>
      </form>
    </body>
</html>
```

Ora salvate il tutto in: *modulo.html*.

Lo script vero e proprio in PHP

Apriamo il nostro editor di testo e digitiamo:

```
<?
$msg="e-mail inviata dal sito\n";
$msg.="Nome del mittente:\t$nome_mittente\n";
$msg.="Indirizzo e-mail del mittente:\t$mail_mittente\n";
$msg.="Messaggio:\t$messaggio\n\n";
$destinatario="tu@tuaemail.it";
$oggetto="Feedback del sito";
$intestazioni_mail="From: il mio sito<>\n";
$intestazioni_mail.="Reply-to: $mail_mittente\n\n";
mail($destinatario, $oggetto, $msg, $intestazioni_mail);
?>
```

Ora aggiungiamo qualche riga in HTML che ci permette di sapere se veramente l'e-mail è stata inviata.
Nello stesso file continuiamo a scrivere:

```
<html>
<head><title>E-Mail inviata</title>
</head>
<body>
<h1>La tua e-mail è stata inviata con successo</h1>
</body>
</html>
```

Ora non ci resta che salvare il documento nella stessa cartella del precedente file (modulo.html) in: invia_modulo.php. Ora trasportiamo entrambi i file nella cartella del server php, e facciamo partire con il browser il file modulo.html, inseriamo tutti i capi e premiamo *OK*...
...Se tutto è andato bene il modulo ci risponderà positivamente. In pochi passi abbiamo realizzato un pagina web dinamica utilizzando il PHP che invia una e-mail a un destinatario. È normale che questo script possa sembrare un po scarno ma è un ottimo modo per iniziare a creare una pagina completa per l'invio di e-mail. Infatti con semplici aggiornamenti è possibile inserirvi altre funzioni di usabilità del codice, un update quasi necessario potrebbe essere sostituire alla nostra e-mail una variabile $destinatario che ci permetta di inviare l'e-mail a chiunque, oppure implementare l'invio di file allegati ecc.

COME LEGGERE UN FILE XML DA PYTHON

Il seguente script usa DOM ma è eventualmente possibile anche usare SAX

```
from xml.dom import utils,core
import string
reader = utils.FileReader('rss.xml')
doc = reader.document
Storage = ""
for n in doc.documentElement.childNodes:
 if n.nodeType == core.TITLE:
  Storage = Storage + n.nodeValue
print len(string.split(Storage))
```

PER UNA VIRGOLA IN PIÙ

Le prime specifiche di C# prevedevano che nelle dichiarazioni di tipi enum, nelle inizializzazioni degli array e all'atto della specifica degli attributi, si potessero usare le virgole solo per separare i vari elementi. In parole povere, non era consentita una scrittura di questo tipo:

```
object[] MioOggetto = new object[] {x, y, z,}
```

La virgola posta dopo il parametro z avrebbe causato un errore durante la compilazione. Ebbene, non è più così. L'istruzione appena accennata è corretta secondo le specifiche di linguaggio di C#. Altrettanto corretta è una dichiarazione di questo tipo:

```
public enum X
{
  A,
  B,
  C,
}
```

Potrebbe sembrare una cosa da nulla, ma diventa di estrema comodità nel momento in cui vogliamo cambiare l'ordine degli elementi indicati.

COPIARE NELLA CLIPBOARD

Fondamentale nelle applicazioni Windows, la clipboard consente di scambiare dati fra le applicazioni nel modo più semplice. Altrettanto semplice è implementare questa funzionalità con VB.NET. Il metodo Clipboard.SetDataObject permette di copiare nella clipboard una stringa, del testo formattato rtf o un'imagine, in modo immediato. L'esempio che segue copia il contenuto di una TextBox, nel caso in cui la TextBox non sia vuota:

```
Sub CopiaTextBox(ByVal tBox As TextBox)
    Dim testo As String = tBox.Text
    If testo.Length > 0 Then
        Clipboard.SetDataObject(testo)
    End If
End Sub
```

INCOLLARE DALLA CLIPBOARD

Speculare all'esempio appena visto, l'operazione di "past" consente di estrarre informazioni dalla clipboard è altrettanto semplice. Bisogna giusto porre un po' di attenzione sul tipo di dati presenti e verificare che siano conformi a quelli accettati dall'oggetto in cui abbiamo deciso di incollarli. Dunque, con il metodo GetDataObject dell'oggetto Clipboard recuperiamo un oggetto IDataObject, su cui invochiamo il metodo GetDataPresent, per testare il tipo di dati contenuti. Il metodo GetDataPresent accetta due argomenti: il primo specifica il tipo di dati rispetto a cui testare l'oggetto, il secondo è una valore booleano che, se settato a true, il metodo tenta una conversione verso il tipo specificato come primo parametro. Se il test effettuato con GetDataPresent avrà dato esito positivo, potremo proseguire con l'estrazione dei dati attraverso il metodo GetData.

```
Sub IncollaTextBox(ByVal tBox As TextBox)
    Dim data As IDataObject = Clipboard.GetDataObject
    If data.GetDataPresent(DataFormats.Text, True) Then
        tBox.SelectedText = data.GetData(DataFormats.Text,
                                                    True).ToString
    End If
End Sub
```

Connettersi a un DB in ASP in ambiente Macromedia Dreamweaver

L'uso dei Database nell'ambito del web è sempre più frequente. Gli ambienti visuali di sviluppo come Front Page e Macromedia sono molto usati per la creazione di semplici pagine html. Accanto alle più frequenti funzioni per la creazione di siti web tali software mettono a disposizione dei programmatori utili tools per la manipolazione di database; ad esempio mediante la metodologia ASP (active server pages). Come tutti i linguaggi ASP può essere programmato con il più spartano degli editor. Innegabili però, sono i vantaggi che si possono avere con l'uso di un ambiente di sviluppo visuale e integrato come Macromedia Dreamweaver. Esso da un lato è ottimo strumento per lo creazione di pagine html, pronte a essere pubblicate sul web, dall'altro si presta alle soluzioni di moltissime situazioni correlate allo sviluppo su piattaforme web. Nel caso specifico integra ottimamente ASP e la tecnologia sottesa all'uso e alla manipolazione di DB da web. Purché abbiate un server web (Pws, Apache o IIS) installato su pc potrete in pochi passi connettere un DB al sito in costruzione. La connessione mediante ODBC è lo stadio preliminare (utilizzando il software in esame). Una volta connesso il DB al server mediante ASP è possibile comandarlo semplicemente programmando. Passo zero per il raggiungimento dell'obbiettivo è aprire il software sopraccitato, si tratta della versione MX 2004.

Fabio Grimaldi

‹1› AVVIO DEL SOFTWARE E CREAZIONE DEL SITO WEB

Dopo aver avviato Macromedia Dreamweaver definiamo un nuovo sito dal omonimo menu. Selezionando gestisci sito appare una nuova finestra di dialogo dove sono presenti tutti i siti che si stanno sviluppando. La lista è vuota se è la prima volta che utilizziamo il software.
Scegliamo la voce nuovo.

‹2› CREAZIONE GUIDATA DELLE CARATTERISTICHE DEL SITO

La creazione guidata del sito e delle sue caratteristiche è accompagnata da svariati passi associati alla stessa finestra di dialogo. Dopo aver immesso il nome del sito (chiamato Libri come l'omonimo DB), cliccando sua avanti viene chiesto se si vuole utilizzare la tecnologia server. Rispondiamo si e selezioniamo ASP javascript.

‹3› L'AREA DI LAVORO COME UNA CARTELLA LOCALE

Lo sviluppo può essere fatto provando direttamente i file su di un server collegato in remoto al nostro PC. Noi optiamo per la strada più semplice ed efficiente in fase di primo sviluppo. Scegliamo la prima voce: modifiche e prove localmente su una cartella a nostra scelta. *C:\inetpub\wwwroot* è il default a cui fa riferimento IIS.

‹4› COMUNICAZIONE TRA L'URL E IL SERVER DI PROVA

La comunicazione tra lURL e il server di prova avviene attraverso il conosciuto protocollo HTTP. Si rende necessario un indirizzo che per default è http://localhost/libri. Manteniamo tale indirizzo e dopo averlo testato con un apposito click su prova URL chiudiamo la procedura di creazione del sito.

‹5› CONFIGURAZIONE DEL DNS (DATA SOURCE NAME)

Dal menu elabora o direttamente sul segno + dell'apposita finestra applicazioni ASP definiamo il DNS. Con questo procedimento si legherà il DB precedentemente creato in Access (ci si può connettere anche ad altri formati di DB).
Dalla voce seleziona scegliere il file mdb che si intende connettere al sito in fase di sviluppo (nel esempio libri.mdb).

‹6› DENOMINAZIONE DELLA CONNESSIONE

Nella successiva finestra di dialogo tra i DNS presenti oltre al DB di prova apparirà quello appena connesso.
Si nomina la nuova connessione LinkLibri e si termina la fase prefissata.
Cliccando sul tasto prova si potrà verificare se l'operazione è avvenuta correttamente.
Adesso il DB è pronto ad essere usato nel sito.
Se avete la pazienza di dare uno sguardo al codice generato, noterete che la vostra connessione è stata trasformata nel corrispondente codice ASP.

Query di aggiornamento in MS Access

Il potente DBMS MS Access mette a disposizione dei programmatori un untile strumento per l'aggiornamento automatico di tabelle o più precisamente deai dati di alcuni specifici attributi. Nell'esempio riportato di seguito si può osservare come, facilmente, sia possibile aggiornare un campo prezzo per realizzare l'aggiornamento da lire a euro. È evidente che se si acquisisce la giusta dimestichezza con tale mezzo si possono rapidamente aggiornare i dati evitando noiosi procedure manuali. Si pensi ad aumenti o ribassi percentuali di prezzi, cambi di parametri di calcolo, operazioni su parti di testo e quant'altro debba essere fatto per tutti i valori assunti da un attributo di una tabella. Il vantaggio si rende ancora più evidente per cospicue quantità di dati. Per l'esempio proposto il DB di riferimento è una libreria che tra le varie tabelle ne contiene una di nome libro. Tra le informazioni che caratterizzano libro vi è il prezzo espresso in lire. Scopo della query di aggiornamento è quello di trasformarlo in euro applicando in automatico la divisione per la conosciuta costante 1936.24. O meglio, trasformare in euro tutti i dati appartenenti al campo prezzo. Lo strumento usato sarà appunto query di aggiornamento. Vediamo come si procede.

Fabio Grimaldi

‹1› TABELLA DI PARTENZA

Libro : Tabella

	IDL	Titolo	Autore	Collocazi	CE	Prezzo
	1	L'ultimo teorema di Fermat	Singh	A03	rcs-libri	20000
	2	Algoritmi + Strutture dati	Wirt N.	D10	Università ·	26500
▶	3	Il linguaggio C++	Stroustrup B.	D11	Addison-W	74000
*	ntatore)					0

La tabella libro contiene una serie di attributi che la caratterizzano, tra questi vi è *prezzo*, di tipo numerico con precisione singola e con un'unica cifra decimale. Dal raccoglitore degli strumenti si seleziona query e si avvia, così, il procedimento per l'aggiornamento del campo prezzo.

‹2› STRUTTURA DELLA QUERY

La query viene creata in modalità struttura. Nella finestra di dialogo che appare, si aggiunge la tabella che contiene i dati che vogliamo aggiornare, ossia, appartenenti a libro.
Successivamente, si selezionano i campi (attributi); nel caso specifico si indica, con un doppio click il solo prezzo.

‹3› QUERY DI AGGIORNAMENTO

Dall'apposito menu, tra le query a disposizione, si sceglie quella di aggiornamento. In metodo alternativo per la scelta fa uso della barra dei menù alla voce: query>query di aggiornamento. Come si potrà notare, nella griglia di struttura della query è apparsa la nuova voce aggiornamento.

‹4› CRITERIO DI AGGIORNAMENTO

Nella casella, incrocio tra la voce aggiornamento e l'attributo prezzo, si deve inserire il criterio di aggiornamento per tutti i dati della colonna attributo prezzo. L'operazione matematica per trasformare le lire in euro è dividere il prezzo (che inizialmente è espresso in lire) per la famosa costante 1936.27.

‹5› ESECUZIONE DELLA QUERY PRODOTTA

Cliccando sul punto esclamativo rosso, si esegue la query di aggiornamento sviluppata.
Prima di produrre il risultato mediante finestre di conferma l'ambiente chiede se si è sicuri di voler realmente modificare in automatico i dati.
Noi rispondiamo si e procediamo.
Chiudendo la struttura si può salvare la query che sarà eseguibile anche successivamente.

‹6› ECCO IL RISULTATO OTTENUTO

CE	Prezzo
rcs-libri	10,3
Università + Rice	13,7
Addison-Wesley	38,2
	0,0

L'ultimo passo è verificare i risultati. Un modo per farlo è ritornare al raccoglitore degli strumenti e selezionare le tabelle, quindi tra le presenti, aprire libro.
Si potrà notare come la colonna dei prezzi è stata modificata nel modo desiderato.
I prezzi sono adesso espressi in euro.
La query di aggiornamento automaticamente avrà aggiornato l'intera tabella secondo le indicazioni ricevute.

Recordset in ASP con Macromedia Dreamweaver

La gestione e la manipolazione di DB da applicazioni su web è un prezioso know how. Macromedia Dreamweaver nella versione MX, consente di realizzare tali mansioni grazie ad un incapsulamento di tutti i metodi più recenti che si occupano della questione. Nell'esempio che esamineremo osserveremo come si possa utilizzare con profitto ASP a tale scopo. Dopo le operazioni: di routine di apertura dell'applicazione; di creazione di un nuovo sito che parli la tecnologia ASP javascript (opzione che bisogna indicare in fase di costruzione); e di creazione di un DSN (data source name) che si agganci al Database precedentemente creato, si passa alla fase successiva. Si vuole creare una vista della tabella libri contenuta nel DB libreria in modo che sia ordinata per titolo, ovviamente il tutto visualizzabile da un browser web. Prerequisiti di sistema sono l'installazione sul proprio pc di un qualsiasi server web come IIS (internet information services) ma va anche bene PWS (personal web server) e di un server di applicazioni, che per gli utenti windows è MDAC (Microsoft Data Access Component). Si può quindi procedere nella creazione e gestione di un recordset che centri l'obbiettivo che ci siamo prefissi.

Fabio Grimaldi

‹1› ATTIVAZIONE DEL RECORDSET

Per cominciare, dai menu riportati nella parte destra dell'ambiente di lavoro selezioniamo la voce *applicazioni* e di seguito *associazioni.* Cliccando con il mouse simbolo più (+) verranno elencati gli elementi che si possono aggiungere alla nostra applicazione dinamica. Tra questi preferiamo il recordset, oggetto della nostra esperienza.

‹2› SCELTA DEGLI ATTRIBUTI

Una finestra di dialogo appare all'utente. Da qui possiamo scegliere gli attributi da aggiungere al recordset, noi optiamo per tutti (ovviamente, se esigenze diverse lo richiedono si possono indicare solo alcuni campi). Dalla scelta multipla in basso indichiamo l'attributo titolo a cui associamo ordinamento ascendente. Con il tasto prova si può già testare la query.

‹3› MODALITÀ AVANZATA

La modalità riportata al punto precedente è la semplice. Cliccando su avanzata vi è la possibilità di una maggiore manipolazione dei dati. Come si può notare, viene associato il codice SQL che realizza la query. Naturalmente, possiamo modificare autonomamente tale codice andando a eseguire una qualsiasi interrogazione SQL.

‹4› COSTRUZIONE DELLA TABELLA

Si inserisce una tabella all'interno della quale (nelle singole celle) si riporteranno i vari campi del recordset appena creato. L'operazione avviene facilmente per trascinamento dei singoli campi all'interno delle celle. La prima riga della tabella conterrà le intestazioni degli attributi. Semplici modifiche sui caratteri e gli sfondi renderanno il tutto più gradevole.

‹5› RIPETIZIONE DEI DATI

Si evidenzia la seconda riga della tabella, dove sono presenti le parti variabili che saranno visualizzate.

Così, potremo indicare quanti record visualizzare. Sempre con riferimento alla finestra applicazioni presente nel lato destro dell'ambiente, si seleziona comportamenti e dal menu relativo opteremo per area ripetuta, che verrà associata alla riga evidenziata.

Questo consente di visualizzare un numero predefinito di record.

Nel caso si volesse visualizzare l'intero contesto del DB è opportuno cliccare su *"tutti i record".*

‹6› LA PAGINA WEB OTTENUTA

Si salva la pagina ottenuta (se non sono stati fatti errori avrà estensione asp).

Siamo pronti per visualizzare i risultati.

Premendo il tasto F12 si ottiene un'anteprima della pagina web creata.

Si attiva il browser predefinito. Il risultato è proprio quello sperato.

Una tabella ordinata per titolo che contiene i dati riferiti al libro.

Utilizzare un file XML come un piccolo database

Si presenta spesso il caso in cui dobbiamo immagazzinare dei dati strutturati su disco ma... non crediamo sia il caso di scomodare un database! Piccole tabelle, log, dati di appoggio, configurazioni: sono tanti i casi in cui la potenza e soprattutto l'ingombro di un DBMS appaiono eccessivi. Eppure le modalità di accesso ai dati che avremmo a disposizione con un database ci tornerebbero utili anche in questi casi: come fare? Un bel file XML pare essere la soluzione ideale. Leggero, portatile, di facile lettura e, grazie alle classi messe a disposizione da C#, accessibile in modo del tutto analogo ad un database.
Vedremo come utilizzare un datagrid per interagire con XML e impareremo sia a interrogare il file XML, sia a modificarlo come se fosse un database.
Non solo: se apriamo il file XML da Visual Studio potremo visualizzare e aggiornare il contenuto di un file XML attraverso un'interfaccia in stile DB che ne semplifica enormemente la gestione.

Raffaele del Monaco

‹1› SCRIVIAMO IL FILE XML

Apriamo il più potente editor XML a nostra disposizione e scriviamo le informazioni che vogliamo immagazzinare, così come indicato in figura. Noi abbiamo usato il notepad di Windows! Salviamo e proviamo a caricare lo stesso file con il Visual Studio. Abbiamo scelto il non originalissimo nome di "base" per il file.

‹2› XML VISTO DA VISUAL STUDIO

Aprendolo con l'editor di Visual Studio, potremo ovviamente approfittare della colorazione sintattica ma, ancora più interessante, cliccando sul pulsante dati indicato dal puntatore, potremo interagire con il nostro file XML esattamente come fosse un DB, aggiungendo o eliminando record, con la garanzia che tutte le modiche si rifletteranno nel file originale.

‹3› IMPOSTIAMO L'INTERFACCIA

Dalla scheda Windows Form della casella degli strumenti, selezioniamo un *OpenFileDialog* e trasciniamolo sulla form, lo utilizzeremo per selezionare il file da aprire.
Selezioniamo e trasciniamo un *DataGrid*.
Trasciniamo anche un *MainMenu*, aggiungiamo la voce *File* e, all'interno di quest'ultima, la voce *Apri*, fino ad ottenere una situazione simile a quella riportata in figura.

‹4› CARICHIAMO I DATI

```
if(openFileDialog1.ShowDialog()==DialogResult.OK)
{
  DataSet ds=new DataSet();
  ds.ReadXml(openFileDialog1.FileName);
  dataGrid1.DataSource = ds.Tables[0].DefaultView;
}
```

Con un doppio clic sulla voce *Apri* ci portiamo nella sezione di codice che gestisce l'evento relativo.
Aggiungendo le righe di codice proposte sopra, potremo caricare il dataset ds con i dati presenti nel file XML e popolare il DataGrid con gli stessi dati. Già con queste poche righe di codice possiamo interagire con il file XML con tutte le possibilità offerte dal datagrid, intervenendo come se avessimo a che fare con le tabelle di un database.

‹5› EFFETTUARE UNA RICERCA

```
public DataRow[] searchForm(string tables,string
                                         expression)
{
  DataTable dt=ds.Tables[tables];
  DataRow[] drs=dt.Select(expression);
  return drs;
}
```

Il metodo che proponiamo in questo codice consente di effettuare delle ricerche sulla tabella passata come primo parametro e utilizzando come chiave di ricerche la stringa immessa come secondo parametro.
Come vedete, possiamo completamente dimenticarci che stiamo avendo a che fare con un file XML. Tutte le azioni sono ormai indipendenti dalla fonte originaria.

‹6› CANCELLARE DELLE RIGHE

```
public bool deleteRows(string tables,string
                                         expression)
{
  bool result=false;
  DataTable dt=ds.Tables[tables];
  DataRow[] drs=dt.Select(expression);
  if(drs.Length>0)
  {
    for(int i=0;i<drs.Length;i++)
              <DRS[].LENGTH;I++)<>drs[i].Delete();
      ds.WriteXml(fileName);
    result=true;
  }
  else {result=false;}
  return result;
}
```

Qui vediamo come cancellare le righe che soddisfano i requisiti specificati come secondo parametro.
Notate il metodo *WriteXml* del dataset che consente di aggiornare il file.

SOFTWARE SUL CD

Apache 1.3.33/2.0.52

Uno dei server Web più usati al mondo

Un'indispensabile ormai. ioProgrammo lo ripropone spessissimo fornendovi sempre le versioni più aggiornate. In questo numero Apache sarà usato per gestire più di un progetto. Se avete bisogno di un server Web per provare le vostre applicazioni Web, oppure da usare in sistemi di produzione, Apache è quello che fa per voi.

Directory: /Apache/

PHP 4.3.10/5.0.3

Il linguaggio di scripting per il web

Ormai PHP lo conoscete tutti, e se non lo avete mai usato, sicuramente vi sarà capitato di accedere a qualche sito web sviluppato con PHP. Saprete dunque perciò che è un linguaggio di scripting particolarmente utilizzato per sviluppare Web Application. Incredibilmente potente, fa della completezza del linguaggio, della facilità di apprendimento, della capacità di integrarsi con applicazioni di dababase, i suoi punti di forza.

Directory /PHP

Python 2.3.4

Un linguaggio orientato agli oggetti con tanto di supporto a classi ed ereditarietà

Viene usato in una varietà di applicazioni. A quanto pare è largamente utizzato ad esempio da Google per lo sviluppo dei loro sistemi. Si caratterizza per la gestione dinamica della memoria, per l'elevata portabilità, per la curva di apprendimento relativamente breve. Un linguaggio di programmazione di cui sentiremo parlare a lungo.

Directory: /Python

MySQL 4.1.7

Il server di database OpenSource più diffuso al mondo

MySQL è un indispensabile. ioProgrammo ogni mese vi offre la versione aggiornata. Si tratta del server di database fondamentale per la maggior parte delle applicazioni Internet scritte in PHP. Ma è difussissimo anche per le applicazioni Standalone e grazie ai nuovi connector comincia a essere usato anche dagli sviluppatori .NET

Directory /Mysql

PHPMYADMIN 2.6.0

Molto probabilmente il Frontend più usato al mondo per MySQL

Ci possono essere un milione di motivi per cui usare un'applicazione Web come frontend verso MySQL, ad esempio quando il vostro provider supporta solo la connessione a localhost è assolutamente indispensabile. PHPMyAdmin tuttavia non è un ripiego a un'interfaccia grafica evoluta, anzi la complessità delle funzioni che esporta lo rende molto spesso preferibile a un'interfaccia standalone.

Directory /PHPMyAdmin

Wamp 5_1.4.3

Prova PHP+MySQL+Apache senza sforzo

Molti di voi saranno stati attratti più di una volta dalla voglia di provare questo trio delle meraviglie. Allo stesso modo molti avranno rinunciato causa la difficoltà di dovere installare un server web, un server di database e il linguaggio PHP. Operazioni non complesse ma che per i neofiti possono rappresentare uno scoglio insuperabile. Wamp nasce appunto per eliminare questo imbarazzo iniziale. Si tratta di un tool che in un sol colpo installa e configura i tre componenti necessari per provare a creare Web Application in PHP. Il tutto con una comoda procedura guidata.

Directory: /Wamp

Xmail 1.20

Il mailserver leggero e affidabile

Di mailserver ormai ce n'è in abbondanza. Tuttavia è importante sottolineare che un mailserver è tanto più qualitativamente elevato quanto riesce a gestire senza troppi scossoni grandi volumi di posta. Xmail ha alcune caratteristiche interessanti. Prima di tutto è leggerissimo. Occupa pochissimi Kb. Nonostante questo è straordinariamente veloce e affidabile. Seconda caratteristica non da poco è OpenSource rilasciato sotto licenza GPL. Se considerate che un mailserver affidabile utilizzabile in ambienti di produzione costa svariati migliaia di euro non si tratta di una caratteristica da poco. Infine data la disponibilità dei sorgenti e data la sua diffusione in ambienti di Hosting, esistono per Xmail un quantitativo straordinario di Plugin che ne estendono il valore aggiunto rendendolo un software dal valore inestimabile.

Directory: /xmail

Tortoise CVS

Per mantenere sotto controllo le variazioni apportate al codice

Tortoise CVS è progetto del mese su SourceForge. Normalmente vengono insigniti di questo riconoscimento solo i tool particolarmente meritevoli. Dopo molti sforzi Tortoise CVS ha spodestato dal trono il suo concorrente WinCVS. Si tratta ovviamente di un tool per la gestione dei progetti che fanno uso del Control Version System. Il Control Version System è un sistema che nasce per mantenere traccia delle modifiche che nel corso del tempo vengono approvate al codice sorgente di un progetto. Si tratta di un tool estremamente utile che

consente specialmente a chi è abituato a lavorare in team di procedere in maniera ordinata e senza conflitti di versione.
Directory: /TortoiseCVS

Sqlite 2.8.15
Il nuovo database Bundled con PHP5

Come molti di voi già sapranno, il nuovo PHP5 ha spostato l'attenzione dal supporto a MySQL a quello verso SQLite. SQLite è un database piccolo e leggero che consente di lavorare su archivi di dati pur non dovendo installare un server. Per molti versi lo si può paragonare ad Access ma le sue caratteristiche lo rendono particolarmente versatile e compatibile con la maggior parte delle web application scritte in PHP. Il software che vi presentiamo è un piccolo client a linea di comando, utile per gestire, creare tabelle e database senza dover passare per PHP.
Directory: /Sqlite

SharpDevelop 1.0.3.1761
L'alternativa a Visual Studio a basso costo

Microsoft senza dubbio ha modificato il suo modo di intendere lo sviluppo delle applicazioni con l'introsuzione della piattaforma .NET. Tuttavia lo sviluppo in tecnologia .NET non è una prerogativa dei soli ambienti targati Microsoft. Esistono delle pur valide alternative a Visual Studio. Se si vuole comunque sviluppare in C# ad esempio oppure in VB.NET, SharpDevelop è una ottima soluzione. Visual Studio rimane sempre un ambiente estremamente completo colmo di caratteristiche che lo rendono unico, tuttavia SharpDevelop rappresenta un'ottima alternativa, economica, potente, affidabile.
Directory: /Sharpdevelop

Mantaray 1.4.1
Java Messaging Service & PHP

Di Mantaray tratteremo abbondantemente in questo stesso numero e, d'altra parte, avevamo già dedicato a questo interessante progetto un articolo introduttivo nei mesi scorsi. Si tratta di un tool che consente di sviluppare applicazioni distribuite in modo molto intelligente utilizzando le API JMS di Java. Sicuramente merita attenzione, perciò vi consiglio di approfondire leggendo l'interessante Articolo di Massimiliano Bigatti in questo stesso numero.
Directory: /mantaray

UN RSS READER CON SHARPDEVELOP

```
XmlTextReader newsReader = null;
  newsReader = new XmlTextReader
                          (m_strFileName);
  while (newsReader.Read())
  {
    if (newsReader.NodeType ==
                  XmlNodeType.Element)
    {
      if (newsReader.LocalName.Equals(
                                 "title"))
      {
        richTextBox1.SelectionFont =
                   new Font("Arial", 11);
        richTextBox1.SelectionColor =
                               Color.Red;
        richTextBox1.AppendText(
                  newsReader.ReadString());
        richTextBox1.AppendText(" ");
      }
      if (newsReader.LocalName.Equals(
                                  "link"))
      {
        richTextBox1.SelectionFont =
                   new Font("Arial", 11);
        richTextBox1.AppendText(
                  newsReader.ReadString());
        richTextBox1.AppendText(" ");
      }
  }
}
```

1 CREA UN NUOVO PROGETTO - Dalla pagina iniziale seleziona *Nuova Combinazione*

2 SCEGLI IL TIPO DI APPLICAZIONE - Scegli *Applicazione Windows* e dai un nome appropriato al progetto

3 DISEGNA L'INTERFACCIA - Scegli *Disegna* in basso, e dagli strumenti laterali trascina un *RichTextBox* e un Bottone sulla form

4 INSERISCI IL CODICE - Clicca due volte sul bottone e incolla il codice come sopra. Per eseguire il tutto clicca sull'icona posta vicino all'etichetta *Debug in alto*

Dev C++ 5 Beta 9
Un editor C++ a basso costo

Dev C++ è un editor distribuito su licenza GPL, come tale non ha costi relativi al diritto d'autore.

Come tutto, o quasi tutto, il software GPL la

ASP

eXtreme Shop eCommerce
Il commercio sul web diventa alla vostra portata.
XeCommerce.zip

NET Memory Profile
Ottimizzatore della memoria per chi programma con .NET
NetMemoryProf.zip

AnyLabel
Per la stampa di etichette con qualsiasi sorgente dati.
anylabel10.zip

ASP

Label.NET
Per la stampa di etichette d'ogni tipo.
labelnet10.zip

r-Access
Tool per amministrare online database remoti.
r-access.zip

VISUAL BASIC

Encoder Wizard
Due componenti ActiveX per codificare e decodificare.
EncoderW2.zip

NET Equation Solver
Un componente per la risoluzione delle equazioni ad una variabile.
EquationSolverNET.zip

Math Max
Cento funzioni matematiche per risolvere anche integrali ed equazioni differenziali.
Eval-MathMax.zip

sua economicità non è affatto sinonimo di scarsa qualità. Al contrartio Dev C++ è uno degli editori più amati ed utilizzati da chi sviluppa in C++. Le caratteristiche sono notevoli. Si va dal Debugger integrato, al project Manager, al Class Browser, al Code Completion. L'insieme di queste caratteristiche, oltre ad una leggerezza innata dell'ambiente lo rende particolarmente comodo da utilizzare per sviluppare progetti C++ anche di grandi dimensioni.
Directoru: /Devc++

JBOSS 1.4
L'application Server per applicazioni J2EE solide

In questo stesso numero di ioProgrammo abbiamo presentato JBOSS Nuke Portal. Realmente JBOSS Nuke Portal è solo una delle tante applicazioni sviluppate sulla base di JBOSS Application Server. JBOSS è un framework completo che mette a disposizione del programmatore J2EE una serie di strumenti tali da rendere le proprie applicazioni estremamente sicure, solide e performanti. JBOSS non è Tomcat ma si appoggia a Tomcat, è in realtà un AS che espone metodi e funzioni che possono essere richiamate per costruire applicazioni solide. In questo numero presentiamo la versione 4.0RC1 che funziona con JDK1.5. L'installazione è, al solito, sufficientemente semplice. Basterà settare JAVA_HOME alla directory di installazione del JDK, entrare nella directory *%JBOSS_ HOME/bin* e lanciare *run.bat*. Infine puntate il browser su *http://localhost:8080*. Noterete che il server è Tomcat che di default viene distribuito con JBOSS che è invece l'application server che genera l'interfaccia che vedete quando accedete a localhost:8080.
Directory: /JBoss

Dev-PHP 2.0.9
Ottimo editor PHP OpenSource

Se state iniziando a sviluppare in PHP avrete bisogno di un editor. Scrivere codice con il notepad può essere un esercizio divertente, ma quando iniziate a scrivere script leggermente più complessi si impone la scelta di passare a un editor più completo. DEV-PHP non solo è completo

HELLO WORLD IOPROGRAMMO, CON NETBEANS

1 AVVIAMO UN NUOVO PROGETTO - Dalla interfaccia principale dell'applicazione cliccare su "New Project"

2 CREIAMO UNA JAVA APPLICATION - Nella dialog box successiva scegliamo "Java Application" e cliccchjamo su next

3 DIAMO UN NOME ALL'APPLICAZIONE - Scegliamo il nome del progetto e la posizione dei file, chiediamo inoltre di creare la classe main

4 DIGITIAMO IL CODICE - Inseriamo il codice personalizzando la classe main come in figura. Da notare le capacità di sintax highlighting

5 LANCIAMO L'APPLICAZIONE - Dall'interfaccia di NetBeans compiliamo il tutto e vediamo il risultato

6 HELLO WORLD IOPROGRAMMO - Il risultato è quello in figura. Nonostante la semplicità si apprezzi l'eleganza di Swing

Mod Asp Dot Net

Per programmare in .NET utilizzando APACHE

Straordinario! Una vera chicca! Quando lo abbiamo provato non credevamo ai nostri occhi. Eppure è vero, questo modulo per Apache 2.0 consente di programmare in .NET utilizzando il server Web più usato al mondo: Apache. La cosa interessante è che il framework di Microsoft NON è emulato, viene proprio utilizzato il .NET FrameWork di Microsoft in tutte le sue caratteristiche. Di fatto il modulo non funziona in ambienti Linux. Si tratta di un'evoluzione importante sia per chi possiede il solo Windows XP Home e vuole provare lo stesso a programmare in .NET senza installare IIS sia per coloro che utilizzano questi tool in modo professionale e non vogliono rinunciare né alla potenza e alla flessibilità di Apache né alle funzionalità offerte da .NET. L'installazione non è complessa ma neanche semplicissima.

1) **Installare il modulo utilizzando l'installer fornito con il cd di ioProgrammo *mod_aspdotnet-2.0.0.msi*.**
2) **Aggiungere ad Httpd.conf la riga**

```
LoadModule aspdotnet_module
        modules/mod_aspdotnet.so
```

3) **Creare una directory al di fuori della root di Apache che utilizzeremo come container per le applicazioni Asp .NET. Nel nostro caso abbiamo creato *c:\StoreCSVS***
4) **Aggiungere ad *Httpd.conf* una riga per l'Handler delle estensioni .NET come segue**

```
AddHandler asp.net asax ascx ashx asmx
 aspx axd config cs csproj\licx rem resources
    resx soap vb vbproj vsdisco webinfo
```

5) **In *httpd.conf* effettuare il *Mount* di una directory virtuale di Apache su quella fisica creata in precedenza**

```
AspNetMount /StoreCSVS "C:/StoreCSVS"
```

6) **Ancora in httpd.conf creare un Alias per la directory in questione**

```
Alias /StoreCSVS "C:/StoreCSVS"
```

7) **In httpd conf garantire i permessi in esecuzione alla directory in questione**

```
<Directory "C:/StoreCSVS">
  Options FollowSymlinks ExecCGI
  Order allow,deny
  Allow from all
  DirectoryIndex Default.htm Default.aspx
</Directory>
```

8) **Infine sempre in httpd.conf mappare tutte le richieste delle pagine ASP .NET verso il framework**

```
AliasMatch/aspnet_client/system_web/
        (\d+)_(\d+)_(\d+)_(\d+)/(.*) \
"C:/Windows/Microsoft.NET/Framework
        /v$1.$2.$3/ASP.NETClientFiles/$4"
<Directory \"C:/Windows/Microsoft.NET
  /Framework/ v*/ ASP.NETClientFiles">
    Options FollowSymlinks
    Order allow,deny
    Allow from all
</Directory>
```

Finito. È meno complicato di quello che sembra e, quando avrete finito, potrete eseguire tutte le applicazioni ASP.NET che volete senza limitazioni, utilizzando Apache e non IIS.

Directory: /mod_aspdotnet

ma anche molto potente.

Dotato di code completion, sintax highlighting, funzionalità di ricerca avanzate ed una serie di tool piuttosto interessanti rappresenta una grande scelta per programmare in PHP. Inoltre è un editor straordinariamente leggero, oltre che gratuito ed OpenSource. Da non perdere!

Directory: /DevPHP

TOMCAT 5.5.4

Il servlet container per Java e JSP

L'idea è molto semplice. Sviluppare in java pagine Web. Ad un primo sguardo, Tomcat, potrebbe sembrare un normale Web Server. Ed in effetti è un normale Web Server! In grado di soddisfare le richieste per qualunque pagina Html. Di fatto però Tomcat offre qualcosa in più, ovvero la capacità di soddisfare richieste per applicazioni Java.

Potrebbe sembrare complesso, in realtà lo è meno di quanto sembri.

Immaginate Tomcat come un grande contenitore al cui interno ci sono altri contenitori ciascuno dei quali rappresenta un'applicazione Java, che richiamata da luogo ad una pagina HTML interpretabile da un browser. Questo consente di sviluppare pagine Web (JSP) utilizzando tutta la potenza della normale gerarchia di classi Java e la sintassi e il linguaggio che qualunque programmatore Java conosce bene.

Directory: /tomcat

JDK 1.5.0

Lo strumento indispensabile per sviluppare in Java

Se siete sviluppatori Java o avete intenzione di imparare a programmare in Java avete sicuramente bisogno del JDK. In questo numero vi presentiamo la versione 1.5.0, rilasciata già da qualche mese e di cui ioProgrammo si sta occupando ampiamente. La versione che vi proponiamo è particolarmente interessante, perchè oltre al consueto JDK contiene in bundled una versione di NetBeans, ovvero un ambiente di programmazione appositamente studiato per applicazioni Java. NetBeans è piuttosto potente, anche se molto pesante. È necessario avere un computer performante per lavorare in tranquillità. E d'altra parte il concorrente "NetBeans" non è certo un mostro di leggerezza. In ogni caso le caratteristiche di Eclipse sono interessanti. Si va dal code completion alla sintassi highliting al refactoring. Sicuramente è un ottimo editor alternativo ad Eclipse ed altamente estendibile grazie ai moduli.

Directory: /jdk1.5.0

Hibernate 2.1.7

Il tool per la persistenza dei dati essenziale per programmare in JAVA

Di Hibernate parliamo a lungo in questo stesso numero. Si tratta di un tool che sta assumendo sempre più una posizione di rilievo nel campo della programmazione Java. Consente di mappare i vecchi database SQL relazioni su Oggetti. In pratica astrae la programmazione SQL consentendovi di trattare anche i Database in modo naturale alla stessa stregua della programmazione ad Oggetti. Si tratta di un tool complesso quanto potente. Se siete dei programmatori Java non potrete fare a meno di installarlo e cominciare a usarlo leggendo l'articolo proposto in questo stesso numero di ioProgrammo.

Directory /Hibernate

Irrlicht 0.7

Sviluppare Giochi 3D potenti in modo semplice

Da quando Alfredo Marroccelli ha iniziato la serie di articoli riguardanti lo sviluppo di Giochi 3D basati sul 3D Engine Irrlicht siamo stati letteralmente sommersi da email di lettori che si sono divertiti a programmare il proprio gioco 3D. Irrlicht è straordinariamente potente, facile da usare, intuitivo. Tanto da meritare uno spazio fisso nelle nostre pagine. Potete stare sicuri che continueremo a parlarne a lungo.

Directory: /irrlicht

Jedit 4.2

Un editor per Java molto leggero

Siete stanchi dei bellissimi, potentissi ma pesantissimi editor per Java? È il momento di passare a Jedit. Non sarà un mostro per quantità di caratteristiche disponibili, ma possiede i requisiti essenziali per programmare in Java! E questo è quanto basta. Sintassi Highliting, compressione/ decompressione del codice suddiviso in classi, facilità di ricerca. Da utilizzare se avete macchine non troppo potenti o se non volete perdervi nei meandri delle opzioni degli editor più complessi

Directory: /jedit

Case Studio 2 2.18

Progettare diagrammi Entità-Relazione

Uno strumento che consente di progettare qualsiasi database attraverso i Diagrammi Entità-Relazione. L'interfaccia grafica consente di avere il pieno controllo del progetto ed è possibile generare il relativo DB per tutti i più diffusi DBMS: Oracle, DB2, MSSQL, Access, Sybase, InterBase, Firebird, MaxDB, MySQL, e PostgreSQL. I dettagliati report in HTML si rivelano di grande utilità all'atto della documentazione di un progetto. Versione dimostrativa limitata nelle funzionalità. Nella nuova versione è stato aggiunto il supporto per MySQL 4.1.*

casestudio.zip

Repro 2.0

Una moviola per fermare il bug!

Repro offre una tecnologia che consente di scovare tutte le cause di possibili malfunzionamenti nei nostri software. Repro consente di registrare e riprodurre tutte le azioni degli utenti, tracciando al contempo tutti gli stream di dati, le risorse utilizzate e una serie di grafici sulle performance dell'applicazione. È così possibile ottenere un report di tutto lo stato del PC al momento in cui il bug fa capolino. Versione di prova valida ventuno giorni.

Repro-20.exe

RC Localize 2.6

Localizza le tue applicazioni

Un aiuto nella localizzazione delle risor-

JBOSS NUKE PORTAL

Come PHP-Nuke ma in Java

JBOSS Nuke Portal segue la stessa logica di PHP Nuke e dei vari Nuke che sono susseguiti ma con la caratteristica importante di essere scritto in Java e di avere come base JBOSS, ovvero l'application Server che così tanto sta facendo parlare di se per le caratteristiche avanzate che lo contraddistinguono. JBOSS Nuke di default fornisce una serie di moduli oltre alla classica gestione degli utenti, dei permessi, l'aggiornamento automatico degli articoli e delle news, viene installato con un forum identico a PHPBB con un modulo per la gestione dei download, con un sistema per l'aggiornamento dei Temi. L'installazione è molto semplice. È sufficiente avere installato un JDK1.4 (Non funziona con la versione 1.5 se non a prezzo di modifiche interne al codice) settare la variabile d'ambiente JAVA_HOME affinchè punti alla directory di installazione del JDK, portarsi nella directory /JBOSS_ NUKE_ HOME/bin e lanciare run.bat. Vedrete scorrere sullo schermo una serie di informazioni. Lo script lancia prima di tutto un sistema JBOSS distribuito insieme al Nuke ed effettua automaticamente il deploy dell'applicazione. Il passo successivo è semplicemente puntare il browser su http://localhost:8080/index.html e cominciare a prendere dimestichezza con il vostro nuovo luccicante CMS. Di default vengono forniti due utenti admin con password admin per l'amministrazione e user con password user per la navigazione consueta. Il database di default è HSQLDB che viene lanciato dallo script iniziale.

Directory: /jbossnuke

INSTALL CREATOR 2.0 BUILD 22

Per costruire installazioni facilmente

Un eccellente strumento per creare installazioni in un batter d'occhio.
È sufficiente seguire l'intuitivo Wizard per ottenere installazioni di livello professionale e che lasciano ampia liberta di personalizzazione allo sviluppatore. Gestione delle icone, del registro di sistema, della directory di installazione e molto altro, sono tutte questioni risolte con grande semplicità da questo prodotto che ha anche la piacevole caratteristica di essere gratuito.

icinst.exe

CREIAMO LA NOSTRA INSTALLAZIONE

1 Appena avviato, Install Creator ci accoglie con un Wizard che permette di definire tutti i dettagli del nostro pacchetto di installazione. La prima richiesta è di fissare la directory di partenza del nostro progetto.

2 Le schermate proposte dal Wizard sono numerose, siamo chiamati a scegliere: la lingua dell'installazione, l'icona associata al programma, disclaimer, directory di installazione e così via. In figura vedete la scelta sul tipo di finestra per il setup.

3 Dopo aver scelto se creare o meno il file di disinstallazione, siamo pronti ad avviare la costruzione del setup con un click sul pulsante fine. Dopo aver completato il Wizard, potremo sempre modificare tutte le informazioni immesse.

se associate ai tuoi progetti software. Indispensabile nel momento in cui vorremo provare a spingere i nostri prodotti in mercati più ampi di quello italiano. Tutte le risorse sono navigabili in un albero che permette di tenere sotto controllo l'intero progetto.
Si può utilizzare al posto dell'Esplora Risorse presente in Visual Studio, che si dimostra spesso insufficiente nei momenti più delicati.
Versione di prova valida 15 giorni.
RCLocalize.exe

EventStudio 2.5

Esplora i più complessi scenari con un potente tool CASE

Un sistema che permette di progettare applicazioni a partire da una descrizione discorsiva dei vari scenari in cui l'applicazione sarà utilizzata. Un importante ausilio nella definizione di diagrammi Use Case, Sequenze e per tutti i diagram mi che tengono conto del woekflow del processo.
EventStudio supera la "semplice" progettazione UML, con il supporto per la progettazione Real-time e con una serie di diagrammi non compresi nelle definizioni base di UML. Versione di valutazione valida 45 giorni.
eventstudio.zip

ConversionTool 4.0 build 6

Importare database Access in SQL Server

Se non vi basta il Wizard proposto da Microsoft, questo tool potrebbe fare al caso vostro. Risolvendo alcuni punti dolenti che Microsoft aveva tralasciato: conversione di viste e procedure, traduzione safe dei campi booleani e soprattutto la conversione (non completamente automatica) del codice VBA in funzioni SQL Server.
Versione dimostrativa.
DBX Conv.zip

Java Class Disassembler 4.0

Disassemblare file Java .class

Un'applicazione Windows che, attraverso una scarna interfaccia visuale, consente di disassemblare il bytecode Java. Suggerito per i più smanettoni. Pieno supporto per le novità introdotte con Java 5.
Versione di prova valida trenta giorni.
jcd.zip

VISUAL BASIC

Stat Max

Calcolo delle probabilità e formule statistiche.

Eval-StatMax.zip

WebCab Hypothesis Testing

Intervalli di confidenza e test di ipotesi per .NET.

WBHypTest.zip

PYTHON

ForgetSQL

Accesso alle tabelle SQL attraverso una semplice classe.

forgetSQL-0.5.1.tar.gz

GFPlus 1.1

Una shell interattiva a riga di comando per chi utilizza Gadfly.

gfsql.zip

Phyton Database Objects

Collezione di oggetti indipendenti dal tipo di database.

PDO-1.3.1.zip

C-SHARPENER FOR VB 1.3

Basta un clic per tradurre da VB.NET a C#

Un add-in per Visual Studio che converte automaticamente progetti Visual Basic .NET in progetti C#. Semplicissimo da utilizzare grazie ad un wizard che guida passo passo durante tutte le fasi del passaggio, consente di convertire qualsiasi progetto: applicazioni Windows, applicazioni ASP.NET, librerie di classi, librerie di controlli e Web Services. Molto interessante la nuova funzionalità che consente di lanciare la conversione da riga di comando: sarà più semplice eseguire operazioni ripetitive. Versione dimostrativa, è possibile convertire solo progetto limitati a 500 righe di codice.

c-sharpenerforvb.zip

Trasformare un progetto da VB.NET a C# con C-Sharpner

1 Nel menu strumenti di Visual Studio .NET, possiamo notare la presenza di una nuova voce: se abbiamo aperto un progetto VB.NET, possiamo cliccare su "C-Sharpner su VB" e avviare il Wizard per la trasformazione.

2 Una volta scelti nome e destinazione del nuovo progetto C#, siamo chiamati a definire solo alcune opzioni "secondarie" come quelle relative ai commenti. Molto interessante è invece l'opzione che preserva i file XML creati durante la conversione.

3 Ancora un paio di clic sul pulsante Avanti e arriveremo alla completa traduzione del nostro progetto. È importante sottolineare che il progetto VB.NET di partenza non è in alcun modo alterato durante il processo di conversione.

PYTHON

PyFileMaker

Integrazione tra FileMaker e Phyton.

PyFileMaker-1.2a3.zip

Pymerase

Generare object model per avere l'output desiderato.

pymerase-0.2.1-src.zip

PHP

PHP Book150

Forse il miglior guestbook disponibile (gratis) in PHP/MySQL.

phpbook150.zip

Advanced Guestbook 2.3.1

Tante funzioni originali per uno meccanismo classico.

gbookphp.zip

Access Stats 1.12

Per tenere traccia di tutte le attività occorse sulle vostre pagine web.

stat.zip

Census SB 6.0

Traccia i bug e i difetti dei tuoi proggeti via Web

Un sistema per il tracking dei bug che si presta ad essere utilizzato in ambiente distribuito grazie alla efficace interfaccia Web.

Tra le funzioni salienti ricordiamo la notifica via e-mail, la possibilità di essere fortemente personalizzato ed un'interfaccia particolarmente semplice e intuitiva.
Versione dimostrativa.
censussb.exe

EditLive for Windows 3.5

Integra l'authoring Web nei tuoi siti

Un interessante tool che consente di pubblicare, siti aggiornabili da remoto, attraverso delle operazioni visuali. L'SDK consente di integrare nei nostri siti delle avanzate capacità di authoring: aggiornarli e disegnarne l'interfaccia sarà un'operazione rapida e alla portata di tutti.
Versione di prova valida trenta giorni.
editlivesdk.exe

Java Launcher 2.2

Lanciare applicazioni Java con un doppio clic

Un tool di rara semplicità che consente di distribuire applicazioni Java, garantendo all'utente la massima semplicità nell'avvio. Java Launcher comprime tutte le classi e le risorse relative ad un'applicazione in un unico file .EXE che potrà essere lanciato come una comune applicazione Windows.
Gratuito.
javalauncher.zip

Easy Java 1.2

Fai pratica con Java con un editor amichevole

Per imparare Java è bene non affidarsi da subito a ingombranti IDE: semplificando troppo la stesura delle applica-

zioni, rischiano di nasconderne il funzionamento a chi ha invece bisogno di capire. Dunque, un piccolo editor e la riga di comando saranno i nostri più fedeli alleati nelle prime esplorazioni del linguaggio. Purtroppo, anche in questo caso, ci sono alcuni problemi: chi inizia ha spesso difficoltà a compilare e ad avviare le applicazioni.

Ecco che Easy Java ci viene in soccorso. Il suo editor offre poco più del notepad di Windows, in più ha due piccoli tasti che possono cambiarci la vita: compile e run. Da non sottovalutare la possibilità di fornire gli argomenti dell'applicazione, simulando i parametri passati dalla riga di comando.
Ultima buona notizia: è gratuito!
ezjava_setup.exe

JDebugTool 3.8

Un debugger per Java full optional!

Un debugger stand alone costruito sulla base della JPDA, Java Platform Debugger Architecture. L'interfaccia grafica ed un Help ottimamente strutturato consentono un rapido utilizzo del tool.

Tra le funzioni segnaliamo: debug remoto, debug multi-thread, modifica delle variabili "al volo", visualizzazione delle classi attualmente caricate, valutazione dei *toString*, monitoraggio dell'occupazione di memoria, sincronizzazione con gli eventi di load e unload delle classi.
In questa versione si apprezza particolarmente la velocità di esecuzione.
Demo valida 15 giorni.
jdebugtool_sdk14.jar

UltraEdit-32 10.20d

HTML, testo, esadecimale: tutto in un editor

UltraEdit-32 è principalmente un editor esadecimale con un completo supporto per le macro e numerose funzioni avanzate come la conversione i file da DOS a Unix.

In questa nuova versione sono state aggiunte delle comode funzionalità di autocompletamento, oltre ad un miglioramento complessivo dell'interfaccia, inoltre è ora disponibile il supporto per *SFTP (Secure FTP Support)* ed il *sintax highlighting* che si adatta automaticamente al tipo di documento, basandosi sul nome del file.
Licenza di valutazione valida 45 giorni.
uedit32.zip

JetBrains ReSharper 1.0.5

Un assistente per C# in Visual Studio

Un add-in per Visual Studio .NET 2003 che fornisce un aiuto "intelligente" durante la scrittura di codice C#. Oltre a fornire il rilevamento in tempo reale degli errori di battitura, ReSharper fornisce anche la soluzione più corretta al nostro problema.
Molto interessanti anche le funzionalità

PHP

Top Download 2.0

Questo script conta quante volte un file è stato scaricato.
topdl.zip

Free link page 1.2

Come permettere agli utenti di aggiungere link alle vostre pagine web.
links.zip

PERL

Weblog 2.0

Creare, editare e organizzare canali semplicemente.
tutorial22.zip

Simple AD

Semplice script per la rotazione dei banner.
simplead.zip

Lyles Banner Rotator

Sistema completo in Perl per la gestione dei banner.
lylesbanner.zip

WWW Adverts

Gestione completa dei banner.
wwwadverts.zip

Calendar Script

L'ultimo nato in fatto di calendari di eventi.
calendarscript321.zip

VintaSoft Twain ActiveX Control

Per la gestione completa di scanner e webcam.
VintaSoftTwain21.zip

MSDE MANAGER 3.09

Per gestire database MSDE attraverso una intuitiva interfaccia grafica

Tutte le più comuni operazioni per la gestione di un database MSDE possono essere effettuate per via visuale grazie a MSDE Manager. Sarà possibile aggiungere, modificare ed eliminare qualsiasi elemento: database, tabelle, viste, stored procedure, dati degli utenti e tutto quanto è necessario alla vita di una base di dati. È possibile schedulare le attività più complesse e gestire tutte le fasi di back-up e restore. Molto interessante la possibilità di copiare un database da un server all'altro attraverso un Wizard. Versione di valutazione valida quattordici giorni.
msde.exe

SmartCode ViewerX VNC

Controllo ActiveX per l'utilizzo completo del VNC.

SmartCodeViewerX.exe

CS Printing Engine 1.2

Componente COM motore delle nostre stampe.

CSPrintEngine12.zip

CS HTML DiffControl 1.5

Per "allineare" documenti HTML e XML.

CSHTMLdControl.zip

GigaSoft ProEssential

Centinaia di funzioni per testo e grafica.

GSProEss5.zip

My Generation

Veloce generatore di codice per interfacce per database.

MyGeneration.zip

di refactoring offerte dall'addi-in. Versione di prova valida trenta giorni.
jetbrain.zip

BBC BASIC for Windows 3.12c

Una nuova reincarnazione del Basic

Il BBC BASIC nasce con l'intento di combinare la semplicità del primo Basic con le caratteristiche di flessibilità tipiche dei moderni linguaggi strutturati. Offre il pieno supporto per l'interfaccia grafica di Windows ed include un compilatore assembler per processori 80x86. Versione di prova, la dimensione delle applicazioni è limitata a 8K e non è resa disponibile l'opzione di compilazione.
bbcwdemo.exe

Inno Setup 5.0.6

Installazioni professionali a costo zero

Assolutamente gratuito, Inno Setup ci aiuta a creare dei perfetti pacchetti di installazione per il nostro software. Tra le caratteristiche: interfaccia in stile Windows 2000, possibilità di compattare tutta l'installazione in un singolo exe, uninstaller, compressione, pieno supporto per l'installazione di risorse condivise come gli OCX e altro ancora. Collegandosi al sito dell'autore, è possibile scaricare il completo codice sorgente in Delphi.

Tra le novità segnaliamo la possibilità di definire uno script per la procedura di disinstallazione ed un miglior rapporto di compressione dei file di installazione.
isetup-5.0.6.exe

JURTLE 1.7

Uno strumento didattico davver ben fatto e che prende le mosse dalla famosa tartaruga (turtle) che a molti di noi insegnò i rudimenti del logo sul glorioso Commodore 64

Jurtle è un ambiente di sviluppo integrato al cui in terno sono già presente dei piccoli applicativi Java che effettuano delle semplici evoluzioni sullo schermo. Siamo invitati a modificare questi programmi e a vedere l'effetto di queste modifiche: gli esempi sono sedici, di difficoltà crescente, e questo tipo di apprendimento (imparare "facendo") risulta perfettamente in linea con lo stile di ioProgrammo. È possibile sperimentare anche delle semplici interfacce, imparando a conoscere librerie fondamentali come Swing e Awt. Un ambiente pensato (forse) per i più piccoli ma che consente anche ai "grandi", che vogliono avvicinarsi alla programmazione Java, un approccio più semplice e divertente. Richiede che sia installata una versione della virtual machine pari o superiore alla 1.3. Versione di prova valida sette giorni.

Jurtle_1.7.exe

LA MIA TARTARUGA

1 **Il breve codice che vedete riportato in figura è sufficiente a muovere la nostra tartaruga e farle disegnare un quadrato. Ovviamente, tutto ciò è reso possibile dalla libreria com.otherwise.jurtle.Turtle importata all'inizio del programma.**

2 **Per avviare l'applicazione è sufficiente premere il tasto play presente nella barra degli strumenti. L'interfaccia a schede permette di passare con un semplice click dal codice alla finestra di output. In figura potete ammirare il nostro quadrato.**

3 **È importante notare che è possibile fare interessanti esperimenti anche con le interfacce grafiche attraverso swing. Insomma, non avete più scuse per non imparare Java: installate Jturtle e cominciate a sperimentare!**

AwStats 6.2

Le statistiche del tuo sito a portata di click

AwStats è un completo sistema per la generazione di statistiche accurate per la valutazione degli accessi verso un sito Web.
Riteniamo che per la validità del prodotto debba essergli dedicato uno spazio adeguato.
AwStats è OpenSource e scritto in Perl, tuttavia ha caratteristiche paragonabili a sistemi commerciali del costo di svariate migliaia di euro. Consente di visualizzare statistiche relative alle page view, ai visitatori unici, ai motori di ricerca, ai browser, alle pagine più visitate. Consente di dividere le statistiche per mese, per anno, per giorno, visualizzare report di accesso per fascia d'oraria e ancora molto altro. Caratteristica molto importante di AwStats è che può essere installato su qualunque sistema che supporti il Perl, perciò potrete usarlo con la stragrande maggioranza dei provider di hosting attualmente disponibili. Unico requisito richiesto è l'accesso in lettura ai log del sito web e la possibilità di eseguire cgi-bin in formato perl dalla directory su cui analog è installato.

Directory: /awstats/awstats-62.exe

INSTALLAZIONE STANDARD

Quella che vi mostreremo qui è un'installazione standard su un sitema casalingo basato su Apache e Win32, chiaramente potete estrapolare da questo tutorial solo le parti essenziali per utilizzare Analog su un provider hosting.
All'interno del CD di ioProgrammo nella directory *AwStats* trovate il file *awstats-62.exe.* È sufficiente cliccare due volte sull'eseguibile.
I file verranno copiati nella directory *c:\programmi\AwStats*. In particolare i file .pl e gli altri file di configurazione saranno copiati in *AwStats\wwwroot\cgi-bin.* Potrete decidere di utilizzare questa directory oppure copiare tutto il necessario in un altro punto dell'hard disk. Non è sempre buona norma dare l'accesso in esecuzione via web a software collocati nella *c:\programmi.* Solo per brevità sceglieremo di utilizzare il percorso standard, nelle vostre installazioni in fase di produzione vi consigliamo di spostare i file in questione in una directory più sicura.

CONFIGURAZIONE DI APACHE

Editate il file *httpd.conf* nella directory *c:\programmi\Apache Group\Apache2\Conf* e aggiungete le seguenti linee

```
AddHandler cgi-script .cgi .pl
ScriptAlias /stats/ "C:/Programmi/AWStats/
                      wwwroot/cgi-bin/"
<Directory "C:/Programmi/AWStats/
                      wwwroot/cgi-bin/">
    Options FollowSymlinks ExecCGI
    Order allow,deny
    Allow from all
    DirectoryIndex awstats.pl
</Directory>
```

Con questa configurazione comunichiamo ad Apache che i file con estensione *.pl* devono essere eseguiti e non downloadati, settiamo una directory virtuale *Stats* mappata sul percorso di installazione di *AwStats* e facciamo in modo che non sia necessario richiamare il file *awstats.pl* che viene sottinteso in assenza di una qualunque altra chiamata effettuata dal browser alla directory *Stats*.

CONFIGURAZIONE DI AWSTATS

Dando per scontato che abbiate installato l'ActivePerl nella directory *c:\perl*, modificate la prima riga di *awstats.pl* come segue:

```
#!/perl/bin/perl.exe
```

nessun'altra configurazione è necessaria all'interno di questo file. Tutte le altre configurazioni possono essere fatte tramite un file awstats.conf posto nella stessa directory che contiene awstats.pl. Esiste già un modello awstats.conf.model nella directory in questione, quindi è sufficiente rinominarlo e procedere alle configurazioni personalizzate.
In particolare le linee importanti sono:

```
LogFile="C:/Programmi/Apache Group/
                 Apache2/logs/access.log"
```

deve puntare alla directory che contiene i log di Apache

```
LogFormat=4
```

Informa awstats che la tipologia di Log è quella di Apache

```
SiteDomain="localhost"
```

Sostituite a localhost il nome del vostro dominio

```
AllowToUpdateStatsFromBrowser=1
```

Consente di fare l'aggiornamento delle statistiche direttamente dal browser

```
Lang="It"
```

Setta l'interfaccia per il linguaggio italiano.

Infine copiate il contenuto della directory icon all'intero della directory icons contenuta nella root del vostro Web Server Apache. Quando avete terminato le vostre configurazioni riavviate apache e puntate il browser su http://localhost/stats, magicamente vedrete comparire la pagina con le statistiche. Cliccate su aggiorna e avrete tutte le informazioni che avete sempre sognato di ottenere con il vostro sito web.

C#

InaByte CryptCompress .NET

Per disporre di compressione e crittografia nel vostro progetto .NET.

ICryptCompress.zip

ASP MultiChat .NET 2.2

Con solo l'HTML, progettato specificatamente per le chat.

ASPMChat22.zip

InaCalc .NET

Potenziare i fogli di calcolo con nuove formule.

IcalcNET.zip

JAVASCRIPT

Cambiar colore allo sfondo

Come cambiare colore allo sfondo con un click del mouse.

coloresfondo.zip

Osservazione dei sistemi operativi e delle loro funzioni

Scheduling di sistema

Uno dei compiti più importanti che il kernel del sistema operativo è chiamato ad assolvere, è lo scheduling dei job, ovvero la scelta dei processi che dovranno beneficiare dei preziosi servizi della CPU

Utilizza questo spazio per le tue annotazioni

L'altro giorno sistemavo la mia biblioteca, niente di eccezionale, un paio di scaffali, quando mi sono imbattuto in un libro che ritengo fondamentale: *"Operating system concepts"* di Jim Peterson e Avi Silberschatz. Ho cominciato a sfogliarlo e mi sono piacevolmente assorto tra le varie tecniche di gestione dei sistemi operativi. Pur non essendo un sistemista, mi hanno sempre attratto gli algoritmi che il sistema operativo utilizza per la gestione: della cpu, della memoria, del file system e delle altri componenti. Li ritengo, oltre che utili per la comprensione del sistema operativo (il software più importante perché indispensabile), molto interessanti giacché attuano strategie di gestione che potremmo ritrovarci anche in differenti ambiti dalla programmazione. Il giorno successivo un rapido colloquio con il mio editor mi ha confermato anche la volontà redazionale di dedicare un po' di spazio della rivista all'argomento. Tra le pagine di soluzioni ci occuperemo di sistemistica, ma con particolare attenzione all'aspetto implementativo, andando a spulciare tra i vari algoritmi usati nei diversi moduli di gestione del sistema operativo, quelli più interessanti. Non sarà, quindi, una trattazione esaustiva sui sistemi operativi (d'ora in avanti SO), bensì una serie di articoli assestanti per l'analisi di particolari compiti e della relativa soluzione sviluppata dai SO. Lo scheduling è il paradigma più appropriato per descrivere la mini serie, esso rappresenta un insieme di regole, sia per la gestione SO, sia che per l'organizzazione generale di altre attività, come ad esempio quella quotidiana di un lavoratore.
A proposito, personalmente penso che se si dovessi salvare solo pochi libri di informatica, oltre al sopraccitato sceglierei *"Basi di dati e basi di conoscenza"* di J.D.Ulman e *"Algoritmi + strutture dati = programmi"* di N. Wirth.
Certo, questo vale solo se lo zaino a disposizione fosse molto piccolo, visto che i libri che hanno segnato l'evoluzione della materia sono davvero molti.

COSA SONO I PROCESSI

Punto di partenza per l'analisi degli algoritmi di scheduling sono i processi. Il processo è l'elemento fondamentale con il quale il SO, anzi meglio il kernel del SO, ha a che fare. Un *processo* è l'immagine di un programma in esecuzione. Così come un *batch job* è un *processo*. Nella terminologia comune i due termini: *processo* e *job* sono intercambiabili anche se il secondo è maggiormente legato alle attività proprie dei particolari SO conosciuti come sistemi *batch*.
Altri sinonimi di processo che si incontrano a seconda del SO che si usa o dei linguaggi di programmazione che lo manipolano, sono: task, programmi, utenti e attività. Tralasciando programmi e utenti che solo in rari casi hanno lo stesso significato, ci serviremo invece dei termini task e attività come sinonimi di processi.
Un programma se lanciato più volte produce più processi (che evidenzia come i due elementi siano in sostanza diversi). Il programma può essere visto come l'archetipo del processo. I SO che stimolano il nostro interesse sono i multiprogrammati, ovvero quelli che gestiscono l'esecuzione "simultanea" di più task sfruttando le risorse a disposizione quali CPU, memorie e periferiche. L'esponenziale sviluppo delle prestazioni dell'hardware, in termini di velocità di CPU o di quantità di memoria, hanno fatto si che i sistemi si trasformassero da mono-programmati a multi-programmati. Si è passati, in modo naturale, dalla prima alla seconda modalità di sistemi per la concomitante esigenza di sfruttare al massimo le risorse disponibili e per massimizzare le prestazioni delle singole applicazioni in termini di tempi (logiche di ottimizzazione di sistema). Le attuali CPU come noto, riescono a svolgere svariati

miliardi (anche migliaia) di istruzioni nel solo intervallo di un secondo. Con un semplice esempio si può dimostrare come con tale potenza la mono programmazione (l'esecuzione di un solo programma per volta) risulterebbe un enorme spreco. Supponiamo di avere a che fare con un'applicazione fortemente interattiva (che abbia molteplici esigenze di I/O). È noto che tra una operazione di I/O e un'altra trascorrono mediamente tempi che comparati a quelli della CPU sono "enormi". Anche il relativamente breve intervallo che intercorre tra due successive digitazioni di tastiera corrisponde a milioni di cicli macchina (istruzioni compiute dal microprocessore). Così, le operazioni di CPU si concluderebbero in tempi molto brevi ed il processo sarebbe destinato ad una continua attesa delle operazioni di I/O. Risulterebbe evidente una sottoutilizzazione della CPU, che resterebbe ferma per una cospicua porzione del suo totale tempo di disponibilità. La soluzione che nel corso degli anni gli informatici hanno progettato prima e realizzato dopo è quella di condividere la potente CPU facendo in modo che in diversi intervalli di tempo più processi ne potessero usufruire. Cosicché, mentre, ad esempio alcuni processi svolgono operazioni di I/O un altro sfrutta la CPU e altri attendono le risorse richieste. Il risultato è quello di sfruttare appieno le risorse, e servire con risparmio economico (poiché si usa un solo elaboratore) più job seguendo precise regole di ottimizzazione. Si tratta però di stabilire delle strategie per "condividere" le risorse del sistema, tenendo sempre presenti gli obiettivi di progetto e le prestazioni previste. Per capire come procedere entriamo nel merito della questione. Un processo può trovarsi in uno tra quattro stati:

- **Running:** quando occupa la CPU per l'attività di elaborazione;
- **Ready:** se è in attesa della CPU occupata da un altro processo;
- **Waiting:** qualora è in attesa di un dispositivo di I/O;
- **Parking**: se si trova temporaneamente inattivo ed è stato caricato su memoria di massa.

Il sopraccitato J. Peterson distingue anche i due casi di allocazione e terminazione del processo che danno luogo rispettivamente a due distinti stati (*new* e *halted*), che però non considereremo poiché si tratta di condizioni limite in cui si trova il processo e che non intaccano tutte le teorie sullo scheduling che ci apprestiamo a esaminare.

Constatato che la CPU è una risorsa attiva, non divisibile e interrompibile, soltanto un processo se ne potrà servire in un fissato instante di tempo, in altri termini un solo *task* si può trovare in *running.* Tutti gli altri processi che necessitano di CPU occupata si trovano in *ready.* In tale stato un processo possiede tutte le risorse necessarie all'esecuzione tranne la CPU. È il SO che assegna la CPU all'insieme di processi che si trovano in *ready.* Lo scheduling e gli algoritmi relativi definiscono appunto le modalità per decidere quale processo e per quanto tempo dovrà passare dallo stato di *ready* a quello di *running.* Nello stato di *waiting* si attende una risorsa che non sia la CPU quindi una periferica.

Fig. 1: ***Code di waiting e ready***

Quando un processo è inattivo, ossia non produce richieste di CPU per un fissato intervallo di tempo, se è necessario liberare memoria RAM avviene lo *swapping* e il processo viene temporaneamente parcheggiato *(swap out)* in memoria secondaria. Sia in *waiting* che in *ready* i processi si trovano organizzati in code come mostrato in **Figura 1,** per esprimere la transizione di stato di un generico processo.

Esaminiamo come un processo viene descritto usualmente da un SO. La prassi usata è molto generale visto che ogni SO usa le proprie convenzioni. Un record chiamato PCB *(process control block),* mantiene le informazioni necessarie a identificare un singolo processo. In sintesi i campi del PCB sono:

- *Nome del processo (ID);*
- *Priorità;*
- *Stato* (running, ready, waiting o parking);
- *Un puntatore per poterlo connette ad altri PCB;*
- *Program counter* (per conoscere l'indirizzo la prossima istruzione da eseguire)
- Altri campi.

Mediante i puntatori è possibili produrre delle liste per l'implementazione delle code sopraccitate.

Il processo si compone di una parte statica ed una dinamica. Appartengono alla prima parte il nome del processo, e il codice e i dati del programma che lo generano. Sono dinamici invece elementi come lo stato corrente, la priorità, i diritti di accesso e altre informazioni. Da notare che l'assegnazione di alcuni attributi dei processi può avvenire automaticamente da SO o per opera del sistemista (una cono-

SISTEMA OPERATIVO

Una definizione formale di sistema operativo è: l'insieme dei programmi che gestiscono le risorse del computer e fanno da interfaccia tra utente ed elaboratore. Le risorse possono essere hardware, quindi le componenti del computer nonché le periferiche, e software come: file, cartelle, driver, insomma gli elementi di base che i programmi manipolano. Un SO ha un'attività più complessa, ma comunque più interessante se gestisce il multitasking, ovvero più processi richiedono le principali risorse come la CPU, la memoria e le periferiche. In tal caso bisogna attuare delle politiche di gestione che hanno come obiettivo finale quello di migliorare le prestazioni dell'intero sistema.

sciuta figura professionale che si occupa della gestione del SO) che può modificare alcuni elementi da programmazione. Ricordo che alcuni linguaggi come java o C++ e altri più datati ADA o Modula 2 consentono la programmazione concorrente è il controllo dei singoli processi in ambienti multiprogrammati.

NOTA

THREAD

Il thread indica una specializzazione rispetto al task. Infatti, alcuni SO "spezzettano" il task in unità ancora più piccole e parallelizzabili, garantendo ovviamente sistemi di protezione e ricomposizione. Si parlerà quindi di multithreading come metodo per la gestione su un numero finito di risorse di più thread, secondo la stessa logica del multitasking.

MODELLO A MACCHINA VIRTUALE

Un accenno a come sia organizzato il SO è dovuto. Un modo per sintetizzare le sue attività è descritto dal modello a macchina virtuale anche conosciuto come *onion skin*, ossia a buccia di cipolla. I vari strati indicano i moduli di sistema. Quelli più interni sono più vicini alla macchina, mentre i più esterni servono per fornire funzioni user friendly mirate alla semplificazione del lavoro dell'utente. Il cuore della macchina, consentitemi la metafora (va bene anche il cervello), è certamente la CPU, quindi il primo strato del modello, ovvero il nucleo, sarà per l'appunto il kernel. Lo scheduling dei processi è svolto dal kernel, così come l'intera amministrazione della CPU. A seguire i successivi due strati si occupano della gestione della memoria e gestione delle periferiche. Come si può notare man mano che si ci allontana dal nucleo si svolgono compiti che sono più vicini all'utente.

SISTEMI BATCH E ALTRI...

Esiste una classificazione dei SO in funzione dei compiti che esso compie. Un sistema batch ad esempio è caratterizzato da un basso grado di interattività. Tali sistemi prevedono inizialmente la pianificazione di tutti gli input nonché delle operazioni da svolgere. Fatto ciò il sistema si avvia e non richiede, se non in rari casi, ulteriori scambi di informazioni con l'utente. Sono molto diffusi invece, i sistemi diametralmente opposti ai batch ovvero quelli interattivi, in cui è "continuamente" richiesto uno scambio di informazioni con l'utente. Si definisce real time quel sistema in cui deve essere garantito un "plausibile" tempo di risposta che varia a seconda dell'applicazione. Se si tratta del controllo di un'astronave il tempo deve essere bassissimo, mentre se consideriamo la gestione di uno sportello al pubblico è accettabile anche un tempo di risposta maggiore, al limite di un minuto. Sistemi transazionali sono particolari sistemi interattivi multiutente strutturati per gestire quasi totalmente operazioni di updating, ossia aggiornamento di dB, che ovviamente hanno maggiore applicazione in ambiti gestionali.

Dal pesante concentrato al nucleo (gestione hardware) si passa gradualmente al leggero fino ad arrivare all'ultimo strato, l'interprete dei comandi che è puro software (gestione di altro software per renderlo più amichevole). Al centro si trova il file system.

Ogni strato si occupa di risolvere una serie di compiti, o in maniera autonoma o facendo riferimento a primitive contenute nei moduli sottostanti.

LA VALUTAZIONE DELLE PRESTAZIONI

Esistono dei metodi che fanno uso di indici per valutare l'attività e l'efficienza del SO. In questa fase ci interessa sapere come sia valutabile l'attività della CPU. Per un generico intervallo di tempo la CPU può:

- *Servire un processo.* Si parlerà in questo caso di tempo utente;
- *Essere occupata per attività di sistema.* Tempo sistema;
- *Essere inattiva.*

Rispettivamente definiamo i tre tempi: *tu, ts* e *ti*. In particolare *tu* è la sommatoria di tutti i tempi utente della totalità dei processi. Anche gli altri tempi scaturiscono dal totale dei tempi di sistema e di inutilizzo. Il sistema operativo come i singoli task necessita di CPU per una serie di compiti, uno di questi è decidere quale sarà il prossimo processo che passerà dallo stato di *ready* a quello di *running*. Un indice segnala l'operosità della CPU, è conosciuto come percentuale di utilizzo della CPU e si indica con *%CPU*. Il sul valore si può calcolare con la seguente formula:

tt=tu+ts+ti
*%CPU = [(tu+ts)/tt]*100*

tt è la somma dei tempi in gioco. La frazione all'interno della parentesi quadra varia tra 0 e 1. Se il tempo di inutilizzo è minimo, quindi tendente a zero, numeratore e denominatore sono quasi uguali e il loro rapporto è 1, da cui si ottiene una percentuale vicina al 100%. Un valore auspicabile poiché indica che la CPU è sfruttata appieno. Se il tempo di inutilizzo è molto elevato il rapporto produce un valore vicino allo zero, da cui si deduce un'attività molto bassa. Un altro indice focalizza l'attenzione al tempo utente poiché per assurdo una elevata percentuale di utilizzo della CPU potrebbe anche derivare da un intensa attività della CPU da parte del sistema. In tal caso pur a fronte di un buon valore dell'indice non ci troveremmo in una situazione piacevole poiché i processi godrebbero poco della risorsa che verrebbe concessa al solo sistema. Il throughput tiene conto del solo tempo utente.

*THROUGHPUT = (tu/tt)*100*

Per i singoli processi esiste un importante indice che

valuta il totale tempo di attesa per arrivare ad essere nello stato di *running*, come somma dei tempi di *waiting* e *reading*. Tale indice è il turnaround time.

UNA PRIMA SOLUZIONE ALGORITMI FCFS

È arrivato il momento di esaminare alcuni degli algoritmi che i SO utilizzano per attuare lo scheduling. Una prima classificazione distingue due tipi di algoritmi sulla base della politica adottata:

- *Event driven;*
- *Time driven;*

Nel primo caso la CPU viene assegnata ad uno dei processi nella coda di *ready*, per l'esattezza quello in testa e si esegue per intero tale processo a meno che non si verifichi una interruzione esterna. Questa ultima è una richiesta di CPU da parte del SO per eventi esterne; si realizza mediante il metodo delle interruzioni che approfondiremo nel prossimo appuntamento. Nel secondo il rilascio della CPU da un processo a favore di un altro avviene in base al fattore tempo, e non come nel caso precedente quando il job è terminato. Esaminiamo due dei metodi della prima politica la event driven.
Il primo metodo segue la filosofia *FIFO (first in first out)* tipico delle code. L'algoritmo di scheduling è così molto semplice: il primo processo che richiede la CPU viene servito, insomma si scorre in modo ordinato la coda delle richieste *(ready)* senza assegnare alcuna priorità. Tale tecnica è conosciuta come *"first come first served"* o più semplicemente con il suo acronimo *FCFS*. Tale criterio è puramente teorico e non è di fatto adottato poiché soffre di una rilevante controindicazione. Se il primo processo richiede un tempo di esecuzione molto lungo e i successivi tempi molto più brevi, questi ultimi sono costretti ad attendere la terminazione del primo processo. Se si potesse eseguire per ultimo il processo con una richiesta più elevata di CPU vi sarebbe un tempo medio di attesa dei vari processi molto minore (turnaround time). Permettetemi un esempio che riporta lo scheduling ad una attività comune di tutti i giorni, il lavoro di ufficio presso uno sportello, ad esempio quello postale. La coda comprende cinque utenti che richiedono dei servizi postali. Il primo utente deve aprire un libretto di risparmi, tempo stimato 25 minuti, i successivi quattro devono pagare dei CCP, tempo stimato un minuto. Se si rispetta la fila gli utenti 2, 3, 4 e 5 devono aspettare rispettivamente 25, 26, 27 e 28 minuti. Gli utenti sono i processi, l'impiegato è la CPU, l'algoritmo di scheduling è il metodo di scelta. Un miglioramento del metodo è proposto di seguito.

ALGORITMO SJF

Il criterio *"shortest job first"* SJF serve per primo il processo che ha una minore richiesta di tempo di utilizzo della CPU. Il metodo prevede quindi il continuo ordinamento della coda di *ready*. Sulla lista a puntatori di PCB per lo stato di *ready*, ogni nuovo processo (nuovo PCB) verrà aggiunto seguendo un inserction sort rispetto al particolare campo che indica il tempo richiesto di CPU o comunque una stima di esso. Così facendo si migliora il metodo precedente, diminuendo di fatto i tempi medi di attesa dei singoli processi (media dei *turnaround time*). Lo svantaggio verrà riscontrato per i processi che richiedono molta CPU che saranno costretti ad attendere di più per passare in *running*.
Nell'esempio della posta è come se si mandasse in coda alla fila l'utente che deve aprire il libretto di risparmi. Si abbatterebbe il tempo di attesa degli utenti 2, 3, 4 e 5 rispettivamente a 0, 1, 2 e 3 minuti, peggiorando non di molto quello del primo utente. La difficoltà di tale metodo sta nel conoscere il tempo di CPU dei singoli processi. Ad esempio, per scheduling a lungo termine di può usare un tempo limite che è il frutto di stime fatte sulla base di dati consolidati su precedenti esecuzioni dei programmi corrispondenti. In definitiva la stima viene fatta da un sistemista, che però deve stare attento poiché un valore allettante troppo basso, per diminuire i tempi di turnaround, potrebbe provocare un errore di tempo insufficiente all'esecuzione. Per scheduling a breve termine il metodo è stato abbandonato per l'impossibilità di stimare i tempi richiesti di CPU. Un'interessante variante del metodo, che ha invece trovato applicazione è l'algoritmo di scheduling che fa uso di priorità. Ad ogni processo si assegna una priorità scrivendo un valore appartenente ad un prefissato intervallo, nel campo apposito del PCB (di solito gli intervalli usati sono 0 .. 7 oppure 0 .. 4095), e si applica il SJF con le priorità anziché con i tempi. In caso di uguali valori di priorità si procede con FCFS.

CONCLUSIONI

Esistono molti altri metodi, alcuni dei quali saranno i protagonisti della prossima puntata. Necessaria è stata la cospicua trattazione teorica che spero non vi abbia annoiato.
Nei prossimi appuntamenti ci dedicheremo ad altri aspetti della gestione del SO completando dapprima alcuni algoritmi del kernel e proseguendo in ordine con i moduli via via più esterni del modello a macchina virtuale.
Vi aspetto!

Fabio Grimaldi

SCHEDULING A LUNGO MEDIO E BREVE TERMINE

Lo scheduling a lungo termine nei rari casi in cui si adotta è associato alla coda di job associati a un sistema batch. Lo scheduling a medio termine si riferisce alla gestione di processi rimossi dalla memoria centrale. Il più importante è lo scheduling a breve termine che si occupa di individuare tra i processi in ready quello che passerà all'ambito stato di running. La sua strategia è orientata a massimizzare le prestazioni del sistema.

Metodi risolutivi attraverso lo sviluppo di programmi ricorsivi

Quando la ricorsione fa la differenza...

La ricorsione è una tecnica molto utile per il programmatore. Essa è un ottimo grimaldello anche per scardinare le più sicure porte che proteggono ampie casistiche di enigmi

Utilizza questo spazio per le tue annotazioni

Riporto uno dei problemi proposti la volta scorsa: "Si consideri una scacchiera di cinque caselle di lato. Qual è il massimo numero di regine posizionabili nella scacchiera senza che si tengano reciprocamente sotto scacco?" Una generalizzazione è il classico problema delle otto regine, cambia quindi la sola scacchiera che in questo caso è la regolamentare per il gioco degli scacchi. Vi è una vasta casistica di problemi che possono essere risolti con l'ausilio di tecniche di backtracking, il problema delle otto regine è comunque molto esplicativo. Si consideri una scacchiera formata da 64 caselle, disposte su otto righe e otto colonne. Si abbiano inoltre, otto regine (o donne) che come noto è il più potente pezzo che si ha a disposizione negli scacchi, la regina infatti, si può muovere sia orizzontalmente, sia verticalmente e sia in modo obliquo. Il gioco consiste nel collocare le otto regine, ognuna su una colonna diversa, in modo che nessuna risulti sotto l'attacco di una qualsiasi altra. Affinché un regina non sia attaccata da un altra è necessario che non si trovi nella sua colonna, riga o diagonale. Di tale problema si occupò persino il grandissimo matematico C.F. Gauss, nel 1805 anche se egli non diede una soluzione completa, a riprova che è uno di quei problemi che trova naturale soluzione con l'uso dell'elaboratore ma che è di difficile approccio analitico. La soluzione è per tentativi come impone il metodo backtracking è:

Fig. 1: ***Movimento della regina***

```
Procedure aggiungi_regina(col:integer);
ver rig:integer;
Begin
    Rig:=0;  // Inizializzazione
    repeat
        rig:=rig+1;
        if  (posizione (riga,col ) = true) then
        Begin
           colloca_la_regina (riga,col);
           If (col=8) then
           Begin
              visualizza_configurazione;
           esci;
         End;
         Else  aggiungi_regina(col+1);
            (* chiamata ricorsiva *)
            Rimuovi_la_Regina(riga,col);
            (* Ritorno indietro *Backtrack* *)
         End;
         until (riga=8);
End;
```

Per avviare il meccanismo sarà necessario nel main program una chiamata del tipo:

```
Aggiungi_regina(1);
```

che colloca tutte le regine partendo da quella riferita alla prima colonna, ovvero la prima. L'algoritmo inizializza la riga, (uno alla prima iterazione) ed a partire da questo valore verifica se la posizione a disposizione, identificata dalla coppia (riga,col), è accettabile, se è così la regina viene collocata. La funzione booleana posizione(i,j) è vera (restituisce true) se la posizione di riga i e colonna j, non è minacciata dalle altre regine, falsa altrimenti. Si noti che una regina così collocata può essere in futuro rimossa se è necessario. Successivamente, si controlla se le regine collocate sono 8 nel qual caso l'algoritmo è terminato con successo. Basta quindi

emettere in output la configurazione sulla scacchiera delle otto regine. Altrimenti, si aggiunge un'altra regina invocando ricorsivamente la stessa procedura, con un ovvio incremento del parametro colonna. Se si dovesse uscire da una chiamata ad aggiungi_ regina(x) senza aver raggiunto la soluzione finale saremmo di fronte ad un caso in cui non è possibile posizionare una x-sima regina nella x-sima colonna. In questo caso si rimuove la regina precedente, si ritorna quindi indietro, in altre termini si attua il backtracking , che permette l'esame di un'altra soluzione. È qui l'essenza del backtracking. Così procedendo è possibile che vengano rimosse più regine prima di trovare il giusto assetto. La condizione posta al repeat (riga=8) ci consente di esaminare tutte le righe per una fissata colonna.

ANCORA ZUPPA DI PESCE

Merita spazio la soluzione di Roberto Allegra per il problema della zuppa di pesce; cui vi avevo solo accentato la volta scorsa. L'ho apprezzata molto. Ho così deciso di riportare la spiegazione dello stesso Roberto. Ho lasciato anche la prima parte in cui potete notare dei complimenti per il sottoscritto. Scusate ma a volte pecco di vanità. " La tua analisi del problema era perfetta, così come l'idea del set delle utili e delle inutili. Così per prima cosa è stato necessario creare i due insiemi. Il set delle utili viene creato prendendo a caso un insieme di 8 lettere dall'array "ABCD...Z". Il set delle inutili è costituito dall'array degli scarti moltiplicato per due (ovvero, concatenato a se stesso).

Es: *utili="ACEGLOQS", inutili= "BDFHIMNPRTUVZBDFHIMNPRTUVZ").*

Dunque si passa alla scansione del dizionario secondo il primo costraint, ovvero "trovare parole che si incasellino secondo lo schema a tris, per le utili". Il principio che ho applicato è stato di applicare ad ogni casella del tris uno specifico Pattern, fatto di una sequenza di otto elementi 1 e 0 (del resto 8 è il numero perfetto per realizzare bitmask!).

```
{"10010010","01010000","00110001","10001000",
              "01001011","00101000", "10000101",
                        "01000100","00100110"};
```

Questi pattern sono direttamente collegati al set delle utili. Un 1 in seconda posizione, ad esempio, vuol dire che la seconda lettera del set delle utili deve essere presente nella parola una volta sola. Uno 0 indica che tale lettera non deve essere presente nella parola.

Es: utili="ACEGLOQS", pattern="10010010", la parola deve possedere una volta sola le lettere A, G, Q e non deve possedere C,E,L,O,Q, e S. (Questo implica, fra parentesi, che ogni set delle utili genera un set di parole univoco: "ABCDEFGH", ad esempio, genera soluzioni differenti da "HGFEDC BA".)

A questo punto ogni parola che passa il test viene posta in un array[9] (uno per ogni casella) di liste di stringhe.

Le soluzioni generate non sono ancora conformi. Bisogna che le lettere inutili si presentino massimo due volte. Purtroppo, questo significa che la soluzione non può essere lineare, in quanto la scelta di una parola rispetto ad un altra per le prime caselle, condiziona la scelta delle successive. L'unica soluzione è la ricorsione sfrenata!

```
private void FindNextUsableGroup(string tempUnused,
                              int level, string[] path)
{  string myUnused = tempUnused;
   foreach (string T in (ArrayList)
                          patternCompliant[level]) {
    path[level] = T;
    string newUnused = UseWord(T, myUnused);
    if (newUnused != null) {
       if (level==8) {
          ArrayList WordPath = new ArrayList();
          foreach(string S in path)
          WordPath.Add(S);
          Words.Add(WordPath);}
       else {
          FindNextUsableGroup(newUnused, level+1, path);
    } }
   }
}
```

Qui sopra c'è la routine in questione: patternCompliant è l'array[9] delle parole generate dalla fase precedente. Words è l'array in cui vengono memorizzati i risultati. UseWord è una funzione che accetta una parola e una stringa delle inutili (myUnused), e restituisce una nuova stringa delle inutili, in cui sono state tolte le lettere presenti in myUnused.
Se l'operazione non è possibile (lettere non presenti nell'array dell'inutili, ad esempio), la stringa restituita punta a null. A questo punto si ottiene una lista di Array[9], contenenti le varie soluzioni. Da un singolo set di utili possono venire fuori anche 100 soluzioni diverse. E la lista delle soluzioni (univoche!) che il mio programma ha generato in 10 minuti di analisi supera i 5,8Mb (dopodiché ho staccato, prima che mi inondasse l'hard disk). C'è da dire che, alla fin fine, un simile tripudio di soluzioni distinte ruota comunque attorno a un set limitato di parole, dalle caratteristiche ben precise, il cui studio può essere interessante per individuare rapidamente i set delle utili con più probabilità di sfornare soluzioni. Come considerazione finale, penso sia interessante riportare che il mio programma non ha mai trovato (e penso non troverà mai) soluzioni per parole di più di quattro lettere.

CONCLUSIONI

Rinnovo il grazie al nostro collaboratore Roberto Allegra che ha risolto il problema. La soluzione può essere migliorata sottraendo alcuni compiti un po' "rognosi" al lavoro manuale dell'utente e tentando l'ottimizzazione di alcune routine. Queste migliorie potrebbero esser uno spunto per tutta la comunità dei lettori ad aprire una nuova discussione sull'argomento magari sul forum di www.ioprogrammo.it .

Fabio Grimaldi

LE SCATOLE CINESI

Verificare la bontà di una definizione ricorsiva ricorda tanto il gioco delle scatole cinesi. Infatti, si parte di solito dalla regola che intrinsecamente contiene la chiamata ricorsiva e si perviene attraverso diverse chiamate (ognuna interna alla precedente) il cui numero deve essere finito, ad invocare una definizione non ricorsiva che individua un punto di partenza, una sorta di giro di boa. Infatti, risalendo da questo punto a ritroso la catena delle chiamate ricorsive si ottiene la soluzione.

UNA FUNZIONE RICORSIVA

A titolo di esempio ecco una funzione ricorsiva. Implementazione ricorsiva di un sottoprogramma (funzione) per il calcolo del fattoriale di un numero n.

```
function fattoriale(
        x:integer) : integer;
Begin
  if (x=0) then
            fattoriale := 1
  else fattoriale := x*
fattoriale(x-1); (*chiamata ricorsiva*)
End;
```